ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
(INGENIERÍA DE LA SALUD)

# Análisis de expresión diferencial de genes de la planta del tomate infectada con TYLCV

# Differential expression analysis of TYLCV infected tomato genes

Realizado por
**María del Pilar Piñeiro Pereda**
Tutorizado por
**José Manuel Jerez Aragonés**
Departamento
**Lenguajes y Ciencias de la Computación**

UNIVERSIDAD DE MÁLAGA
MÁLAGA, Junio 2017

Fecha defensa:
El Secretario del Tribunal

## Resumen

El objetivo de este proyecto de fin de grado es la aplicación de diversas técnicas para el análisis exhaustivo de expresión diferencial de genes. Estas técnicas se aplicarán a la observación del comportamiento de la especie Solanum Lycopersicum (planta del tomate) ante una infección por el virus del rizado amarillo del tomate (TYLCV), el cual causa enanismo en las plantas, aspecto enredado de la misma y pérdida de vigor así como falta de fructificación.

Para observar el efecto del virus en el comportamiento de la planta, se secuenció y analizó previamente el transcriptoma de la planta del tomate bajo distintas condiciones de infección mediante técnicas de RNA-Seq. Los resultados obtenidos mostraron aparentes inconsistencias en algunas de las comparaciones de la expresión de los genes de la planta y surgió así la necesidad de un análisis temporal de expresión diferencial.

En este proyecto se pretende reunir y poner a prueba los conocimientos adquiridos a lo largo de la formación en este área de la bioinformática. Para ello se repetirá el análisis de expresión diferencial como verificación de los resultados previamente obtenidos y como comienzo de un estudio más complejo para observar la sobre o subexpresión de estos genes a lo largo del tiempo, las funciones de los genes más significativos y las características en común de los últimos.



## Palabras Clave

- Expresión diferencial
- Proyecto de fin de grado
- RNA-Seq
- TYLCV
- Transcriptoma

## Abstract

The aim of this project is the application of a group of techniques for the exhaustive analysis of differential gene expression. These techniques will be applied for the observation of the behavior of the Solanum Lycopersicum species (tomato plant) infected with tomato yellow leaf curl virus (TYLCV), which causes dwarfism in plants, a tangled appearance and loss of vigor as well as lack of fructification.

To observe the effect of the virus on the behavior of the plant, the tomato plant transcriptome was sequenced and analyzed under different conditions of infection using RNA-Seq techniques. The results obtained showed inconsistencies in some of the comparisons of the expression of the plant genes and thus the need for a differential expression analysis over time emerged.

This project aims to gather and test the knowledge acquired throughout the training in this area of bioinformatics. To do this, the analysis of differential expression will be repeated as verification of the previously obtained results and as the beginning of a more complex study to observe the under and over expression of these genes over time, the functions of the most significant genes and the characteristics in common between the last.

## Keywords

- Differential expression
- Final degree Project
- RNA-Seq
- TYLCV
- Transcriptome

# Índice

# 1. Introducción

## Motivación

El análisis de RNA-seq se centra en la identificación de conjuntos de genes que se expresan diferencialmente en diferentes estados biológicos. Existen diversos tipos de análisis, y su rendimiento depende de los datos a los que estos se apliquen.

El problema a resolver en este proyecto, al que se van a aplicar diferentes técnicas de análisis, surgió con la obtención de resultados al efectuar un análisis con DESeq2 en la planta del tomate (*Solanum Lycopersicum*) infectada por el virus TYLCV (*virus del rizado amarillo del tomate o Tomato Yellow Leaf Curl Virus*) bajo distintas condiciones:

- Planta infectada con TYLCV usando mosca (*Bemisia Tabaci*) como vector.
- Planta infectada con TYLCV usando bacteria (*Agrobacterium*) como vector.
- Planta expuesta a mosca (*Bemisia Tabaci*) sin infectar.
- Planta expuesta a bacteria (*Agrobacterium*) sin infectar.
- Planta no tratada (naive).

Al obtener los resultados del análisis, se observó un comportamiento inesperado en los genes de las muestras expuestas a la mosca (infectada y sin infectar).

En la tabla siguiente se resaltan en color las comparaciones en cuestión.

| ***DEGs_padj <0,05*** | ***2dpi*** | ***7dpi*** | ***14 dpi*** | ***21dpi*** |
|---|---|---|---|---|
| Agro_TYLCV/Agro_mock | 321 | 1909 | 3754 | 8417 |
| Bemisia_TYLCV/Bemisia | 1006 | 76 (a) | 4171 | 10422 |
| Agro_mock/Naive | 1970 | 1746 | 270 | 298 |
| Bemisia/Naive | 9181 | 5465 | 0 (b) | 2944 (c) |

*Tabla 1. Resultados del análisis de expresión diferencial realizado por la empresa CNAG.*

En la comparación del grupo de muestras de la planta infectada por mosca (*Bemisia Tabaci*) [1] con el grupo de muestras de la planta expuesta a una mosca sana sin infectar (a), el número de genes diferencialmente expresados a los 2 días disminuye a 76 pasados 7 dpi (días después de la infección) y aumenta significativamente pasados 14 y 21 dpi. Por otro lado, se observa que los genes de las muestras de la mosca sin infectar comparados con los de las muestras control se expresan mucho pasados 2 y 7 dpi y su expresión desciende a 0 pasados 14 dpi (b) para volver a aumentar su expresión pasados 21 dpi (c).

Los cambios observados en la variabilidad de expresión pueden ser causados por la alteración de la dinámica subyacente o por otros factores no directamente relacionados con el virus. Esta variabilidad diferencial es biológicamente importante y por ello se requiere un análisis que se centre en el perfil de los genes y no solo en el número total de sobre o sub expresados.

Este estudio más exhaustivo supondrá un análisis temporal de expresión diferencial en el que se observe la evolución de la expresión de genes a lo largo del tiempo, así como las características comunes entre genes que tengan un patrón similar de expresión. De esta manera, se conseguirá un mejor entendimiento de los procesos e interacciones que tienen lugar desde el momento en el que el virus infecta la planta hasta pasados los 21 días.

## Objetivos

Este Trabajo de Fin de Grado consistirá en la aplicación de técnicas de bioinformática aprendidas a lo largo de los estudios de grado. Se desarrollará un análisis de expresión diferencial exhaustivo aplicado a la reacción de la planta infectada por TYCLV en el que se deberán demostrar diversas aptitudes, desde el manejo de un supercomputador (debido al gran volumen de datos) hasta la correcta interpretación de resultados en el ámbito biológico.

Se pretende así explorar y combinar diversas técnicas de análisis para facilitar el entendimiento del comportamiento del huésped ante el virus. Entre estas técnicas, el análisis temporal de expresión diferencial será clave para observar la corregulación[1] de los genes, que serán agrupados según sus perfiles de expresión a lo largo de las distintas condiciones y de los puntos temporales: 2, 7, 14 y 21 días después de la infección.

## Actualidad

El análisis de expresión génica es la medida de la actividad (de la expresión génica) de miles de genes simultáneamente para crear una imagen global de la función celular. Estos análisis a través de la observación de perfiles permiten, por ejemplo, distinguir entre las células que se están dividiendo activamente, o mostrar cómo las células reaccionan ante una condición del entorno en particular. En muchos experimentos de este tipo se analiza un genoma completo, es decir, cada gen presente en una célula en particular.

Un ejemplo de condición ante la cual una célula reacciona de determinada forma es una infección. “Las infecciones virales normalmente alteran la fisiología del huésped, desviando casi todos los recursos celulares para la producción de componentes específicos del virus, y suprimiendo activamente las defensas del huésped. Como respuesta a la infección, los huéspedes compensan mediante la sobre o subexpresión de determinadas vías y el despliegue de medidas antivirales específicas” [2]. Se conoce que el virus del rizado amarillo del tomate (TYLCV) amenaza la producción de tomate en todo el mundo causando el amarilleo y rizado de las hojas, la ausencia de crecimiento de la planta y la abscisión de las flores, pero la comprensión actual de la respuesta de defensa de la planta huésped a este virus es muy limitada.[3,4,5,6]

[1] Genes que se expresan de forma coordinada.

Mucho esfuerzo ha sido invertido en identificar rasgos celulares que cambian como consecuencia de una infección vírica. Esta tarea ha sido enormemente beneficiada por el desarrollo contemporáneo de tecnologías para el análisis de expresión diferencial. RNA-Seq es una tecnología recientemente desarrollada para el análisis de perfiles de transcriptoma que utiliza métodos de secuenciación avanzados. El uso de RNA-seq para el estudio del perfil de transcriptoma en lugar de microarrays ha disparado el desarrollo de métodos estadísticos para operar con las propiedades de este tipo de datos cuantificables. La medida con RNA-seq de la expresión de genes está basada en el número de lecturas mapeadas a transcritos, lo que resulta en cantidades discretas y distribuciones sesgadas a la izquierda [7]. Por el contrario, las señales de microarrays son intensidades de fluorescencia escaneadas, lo que se traduce en datos de expresión continuos y normales. Los estudios que usan RNA-seq ya han alterado nuestra visión de la extensión y complejidad de los transcriptomas: esta técnica proporciona una medida mucho más precisa de los niveles de transcripciones que otros métodos y ha aumentado la cantidad de datos disponibles, cuyo análisis puede proporcionar mucha información.
El área dedicada al análisis de datos de RNA-seq aún es joven, y nuevos métodos surgen constantemente.

La varianza experimental es un reto importante cuando se trata de datos de secuenciación de alto rendimiento. Esta varianza tiene varias fuentes: replicación de muestreo, replicación técnica, variabilidad dentro de las condiciones biológicas y variabilidad entre las condiciones biológicas. Para estudiar la varianza existen técnicas como ANOVA, “una técnica estadística que evalúa las diferencias potenciales en una variable dependiente a nivel de escala por una variable de nivel nominal que tiene 2 o más categorías” [8]. De esta forma, es posible observar si hay una diferencia realmente significativa entre varios grupos comparados a la vez.

El tiempo es otra variable a tener en cuenta: los experimentos de expresión génica en el curso del tiempo son un método cada vez más popular para explorar procesos biológicos. Los perfiles de expresión de los genes proporcionan una importante caracterización de la función de los mismos, ya que los sistemas biológicos son a la vez de desarrollo y dinámica. Con estos datos es posible estudiar los cambios en la expresión génica en el tiempo y, de este modo, detectar genes diferenciales. Gran parte de los primeros trabajos sobre el análisis de datos de expresión en series de tiempo se basaron en métodos desarrollados originalmente para datos estáticos y, por lo tanto, existe una necesidad de mejorar la metodología. “Dado que la expresión en serie temporal es un proceso temporal, sus características únicas, tales como la autocorrelación entre puntos sucesivos, deben ser incorporadas en el análisis” [9]. Actualmente, se ha desarrollado una técnica en lenguaje R que tiene en cuenta los cambios en el tiempo: “La adopción generalizada de RNA-seq para medir cuantitativamente la expresión génica ha aumentado el alcance de la secuenciación de diseños experimentales para incluir los experimentos temporales. MaSigPro es un paquete de R específicamente adecuado para el análisis de los datos de expresión génica a lo largo del tiempo, que fue desarrollado originalmente para microarrays y que ha incorporado la capacidad de analizar datos de RNA-Seq” [7].

En conjunto, todas estas técnicas de análisis han demostrado un gran potencial para proporcionar información sobre las redes de interacciones huésped-virus, pero aún es

necesario explotar a fondo lo que ofrecen e incorporar una buena capacidad de análisis para dar significado biológico a los resultados.

## Métodos

Este proyecto se dividirá en dos fases:

La primera fase consistirá en un primer análisis de expresión diferencial que se extenderá desde el preprocesado de los archivos .fastq de la secuenciación hasta el propio análisis en R para la comprobación de los datos obtenidos anteriormente.

El diseño del experimento consiste en cinco condiciones anteriormente mencionadas:

- Planta infectada con TYCLV usando la mosca (*Bemisia Tabaci*) como vector.
- Planta infectada con TYCLV usando una bacteria (*Agrobacterium*) como vector.
- Planta expuesta a mosca (*Bemisia Tabaci*) sin infectar.
- Planta expuesta a bacteria (*Agrobacterium*) sin infectar.
- Planta no tratada (naive).

De cada condición se tomaron tres réplicas biológicas[2] en cada punto temporal de un periodo de tiempo pautado en 2, 7, 14 y 21 días después de la infección (dpi).

Las muestras tomadas ya secuenciadas con técnicas de RNA-Seq serán preprocesadas con SeqTrimNext, mapeadas y cuantificadas con STAR y posteriormente analizadas mediante DESeq2, librería de análisis de expresión diferencial de R. Para llevar a cabo este análisis, debido al gran volumen de datos que se deberá manejar, será necesario el uso del supercomputador Picasso de la Universidad de Málaga.

Con la obtención de los resultados y la comparación con los resultados previos concluirá la primera fase del proyecto y se procederá a la segunda fase. Esta segunda fase consistirá en el desarrollo de un análisis que tendrá en cuenta el perfil temporal de los genes. Para ello se aplicarán técnicas de clustering y regresión para observar los genes significativos. De esta forma, se espera facilitar la comprensión del proceso de respuesta de la planta al virus, así como mejorar la caracterización de los genes que intervienen en el proceso de defensa.

[2] Réplica biológica: Consiste en realizar la misma prueba en múltiples muestras del mismo material / tipo de células / tejido.

## Flujo de trabajo. Fases del proyecto

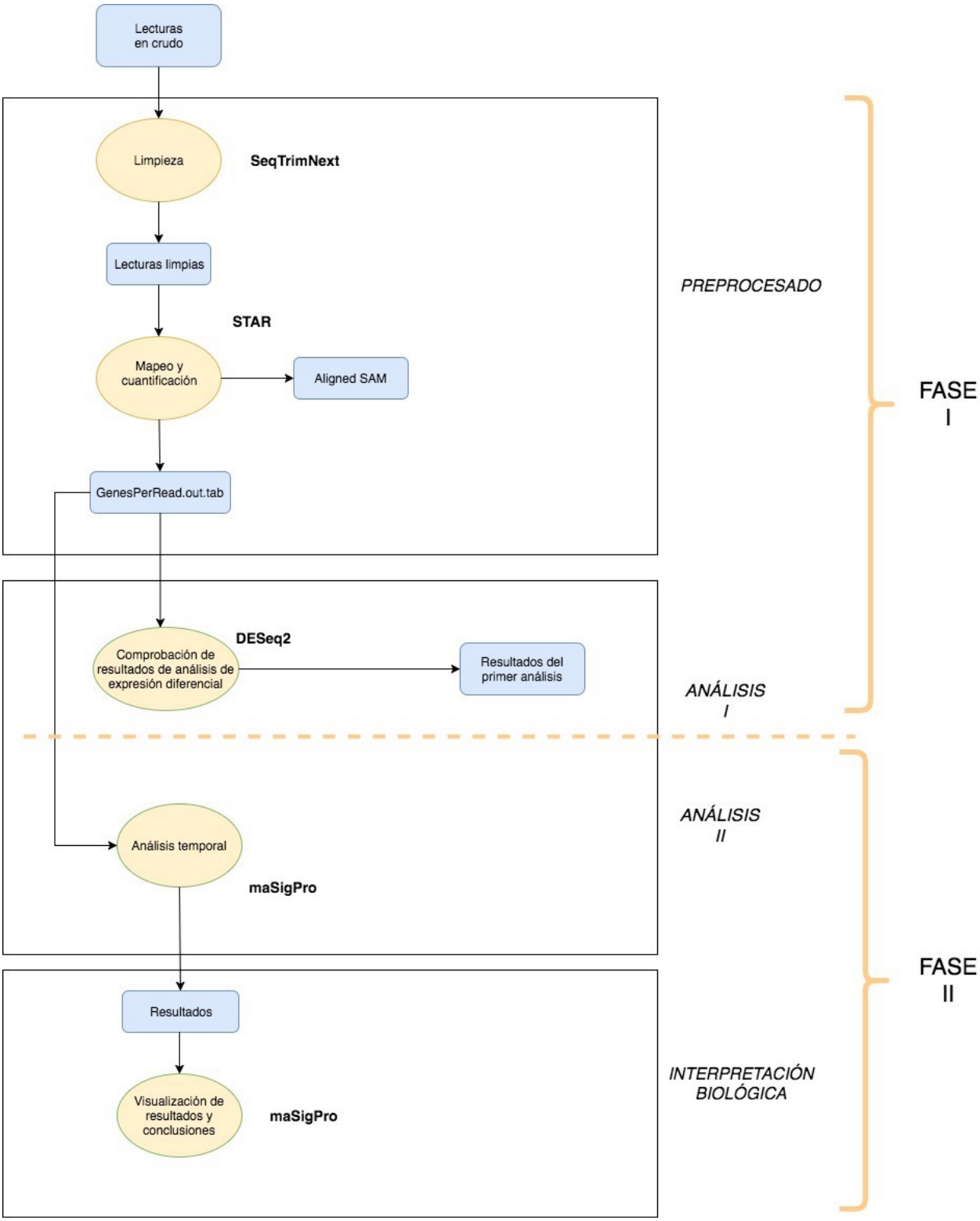



*Figura 1. Flujo de trabajo del proyecto.*

### Fases de trabajo. Planificación

Para el desarrollo del proyecto se aplicará una metodología estructurada en tres fases:

1. Fase inicial de planificación:
    - Reuniones previas de recogida de información y planteamiento del problema
    - Preparación del entorno y decisión de las herramientas a usar
    - Obtención de las lecturas
    - Definición de los pasos a seguir
2. Fase de ejecución:
    - 2.1. La primera fase de la ejecución consistirá en:
        - Limpieza de las lecturas.
        - Mapeo y cuantificación de las lecturas sobre genoma de referencia.
        - Análisis de expresión diferencial y verificación de los resultados previamente obtenidos.
    - 2.2. La segunda fase de la ejecución consistirá en:
        - Diseño e implementación de un análisis temporal de expresión diferencial.
3. Fase final:
    - Conclusiones, discusión sobre los resultados
    - Documentación

# 4. Descripción general del entorno tecnológico

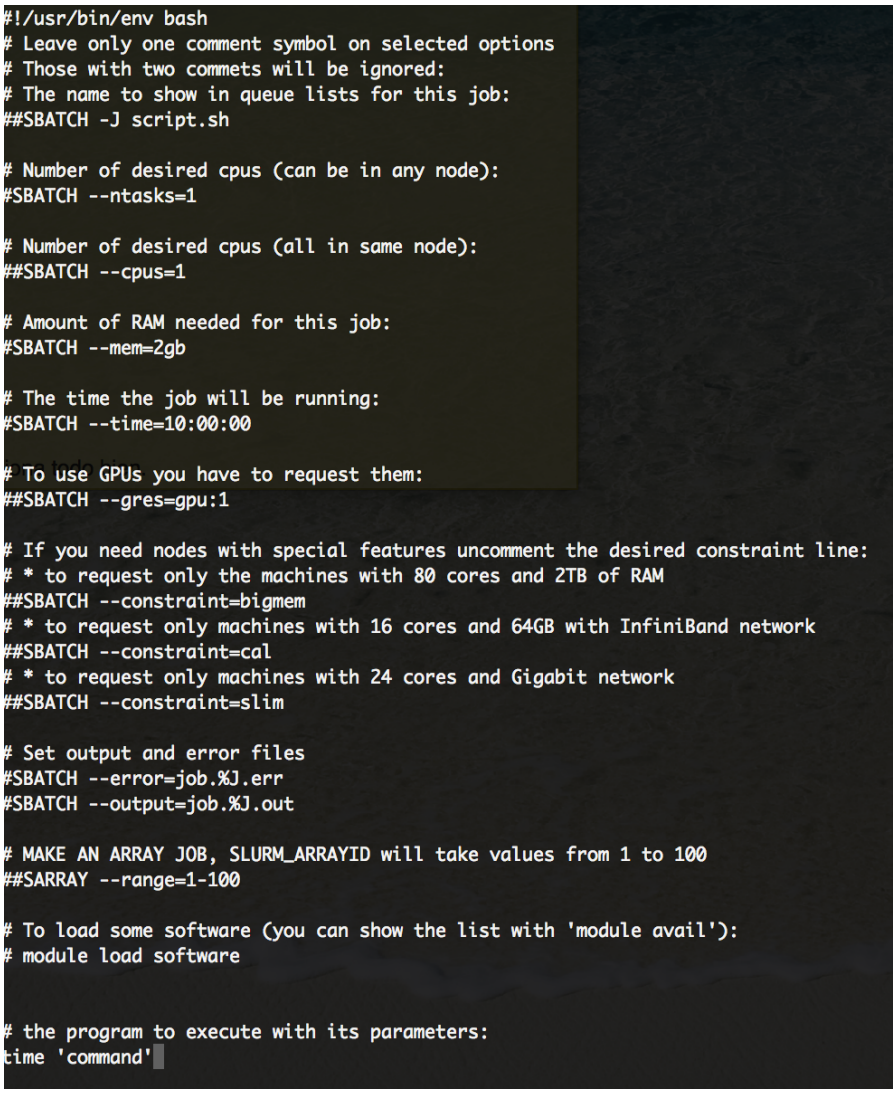
## Tecnologías utilizadas

### Supercomputador Picasso

El supercomputador Picasso del SCBI (Supercomputing and Bioinnovation Center) de la Universidad de Málaga es un conjunto de nodos computacionales con diferentes características.

Todas esas máquinas están unificadas detrás de un único sistema de colas Slurm. Este sistema analiza las peticiones de los usuarios y envía los trabajos a los computadores apropiados para cada tarea. Para enviar un trabajo al sistema de colas se debe escribir un script con un formato específico para pedir los recursos necesarios

que el programa usará (ver figura 2). La transferencia de archivos se realiza mediante conexión SFTP por razones de seguridad. [10]



```
#!/usr/bin/env bash
# Leave only one comment symbol on selected options
# Those with two commets will be ignored:
# The name to show in queue lists for this job:
##SBATCH -J script.sh

# Number of desired cpus (can be in any node):
#SBATCH --ntasks=1

# Number of desired cpus (all in same node):
##SBATCH --cpus=1

# Amount of RAM needed for this job:
#SBATCH --mem=2gb

# The time the job will be running:
#SBATCH --time=10:00:00

# To use GPUs you have to request them:
##SBATCH --gres=gpu:1

# If you need nodes with special features uncomment the desired constraint line:
# * to request only the machines with 80 cores and 2TB of RAM
##SBATCH --constraint=bigmem
# * to request only machines with 16 cores and 64GB with InfiniBand network
##SBATCH --constraint=cal
# * to request only machines with 24 cores and Gigabit network
##SBATCH --constraint=slim

# Set output and error files
#SBATCH --error=job.%J.err
#SBATCH --output=job.%J.out

# MAKE AN ARRAY JOB, SLURM_ARRAYID will take values from 1 to 100
##SARRAY --range=1-100

# To load some software (you can show the list with 'module avail'):
# module load software


# the program to execute with its parameters:
time 'command'
```

*Figura 2. Script tipo para el envío de trabajos a cola.*

Todo el software utilizado en el proyecto está actualmente instalado en Picasso.

## SeqTrimNext v2.0.60

SeqTrimNext es un software distribuido de pre-procesado de secuencias biológicas. Consiste en una adaptación de SeqTrim [11] a secuencias de NGS (Next Generation Sequencing) [12] desarrollado en la Universidad de Málaga.

- STN hace uso de scbi_mapreduce para ser capaz de correr en entornos paralelos y distribuidos.
- Está adaptado especialmente a conjuntos de datos de Roche 454 (normal y paired-end o pareadas) y Illumina, aunque podría adaptarse fácilmente a otras tecnologías de secuenciación.
- SeqTrimNext es muy flexible ya que su arquitectura está basada en plugins que pueden añadirse fácilmente.
- Es capaz de explotar todos los beneficios de un cluster.
- Se proporcionan plantillas por defecto para genómica y transcriptómica, aunque es posible confeccionar una plantilla combinando plugins.

La limpieza de las secuencias es importante antes de mapear las lecturas al genoma de referencia, ya que las secuencias 'crudas' suelen contener lecturas de pobre calidad, restos de secuencias de adaptadores y bases de baja calidad. Aunque esto signifique un menor número de lecturas y mapeos, los resultados serán menos ruidosos.

### STAR v2.5.1b

Mapear consiste en alinear las lecturas (una vez limpias) al genoma de referencia. Este protocolo de mapeo necesita un archivo *.gtf* que contiene las anotaciones del genoma, las cuales permiten a STAR identificar y mapear correctamente los alineamientos.

Mapear grandes conjuntos de datos de secuenciación de lecturas de alto rendimiento a un genoma de referencia es uno de los pasos fundamentales en el análisis de datos RNA-Seq. El paquete de software STAR realiza esta tarea con altos niveles de exactitud y rapidez [13]. Es posible alinear secuencias de cualquier longitud y cualquier tecnología de secuenciación generando archivos de salida que pueden ser usados para muchos análisis como cuantificación de expresión de transcritos/genes, expresión diferencial, visualización de señales, etc.

Tras mapear, si se va a realizar un análisis de expresión, es necesario cuantificar las lecturas de manera que los resultados indiquen el número de mapeos por cada ID de cada gen. Esta tarea ha sido añadida a STAR para que, en el mismo comando de mapeo, se pueda incluir un parámetro para obtener el mapeo cuantificado. Estos resultados una vez cuantificados son los que se analizarán.

### Lenguaje de programación R v3.4.0

Para el análisis de expresión diferencial de los resultados obtenidos se usará el lenguaje R: uno de los lenguajes más utilizados en investigación por la comunidad estadística, siendo además muy popular en el campo de la minería de datos.

Otros motivos por los que escoger R:

- Los entornos de desarrollo son de rápida y fácil configuración.
- Multiplataforma: R puede utilizarse independientemente en cualquiera de los sistemas operativos más utilizados (Windows, IOS y Ubuntu).
- Comunidad muy grande de contribuyentes. R es muy útil para el trabajo interactivo, pero también es un poderoso lenguaje de programación para el desarrollo de nuevas herramientas. [14,15]

Se usará el entorno de desarrollo RStudio Desktop, ya que se trata de una tecnología de Open Source de la que existen gran cantidad de herramientas y multitud de documentación al ser un proyecto colaborativo y abierto [16].

Los repositorios principales de paquetes de R son dos:

- CRAN. Actualmente hay 9000 paquetes en las áreas de finanzas, bioinformática, aprendizaje computacional, computación del alto rendimiento, procesamiento natural del lenguaje…
- Bioconductor. Los paquetes en Bioconductor pertenecen al área de bioinformática. Este repositorio posee reglas con el objetivo de enfatizar la reproducibilidad de los estudios y el constante mantenimiento del paquete [17].

En el caso de que existan paquetes con funciones similares tanto en CRAN como en Bioconductor, es aconsejable utilizar paquetes de Bioconductor debido a la garantía de soporte constante.

### Librerías de R

#### *DESeq2*

El paquete DESeq2 proporciona métodos para analizar la expresión diferencial mediante el uso de modelos lineales generalizados binomiales negativos [18].
Como entrada, el paquete DESeq2 espera datos de recuento obtenidos, por ejemplo, a partir de RNA-Seq u otro experimento de secuenciación de alto rendimiento, en forma de una matriz de valores enteros, ya que el modelo corrige internamente el tamaño de la biblioteca.

Esta librería extrae una tabla de resultados con cambios de pliegue de log2, valores de p y valores de p ajustados.

#### *maSigPro*

Este paquete incorpora el tiempo al análisis de expresión diferencial. Tener en cuenta condiciones y tiempo en el análisis permite observar genes que se corregulan o que se comportan de manera similar. Una nueva adición de maSigPro [19] es la posibilidad

de aplicación a resultados de RNA-Seq, ya que originalmente funcionaba únicamente para microarrays.

El primer paso del método aplica la técnica de mínimos cuadrados para estimar los parámetros del modelo de regresión descrito para cada gen. Este primer análisis genera n-tablas ANOVA, una para cada gen. Un gen que tenga diferentes perfiles entre el grupo de referencia y cualquier otro grupo experimental mostrará algún coeficiente estadísticamente significativo, por tanto su modelo de regresión correspondiente será estadísticamente significativo. El p-valor asociado a la F-Estadística en el modelo de regresión general es el que se utiliza para seleccionar genes significativos. Este valor de P se corrige para comparaciones múltiples aplicando el false discovery rate (FDR).

Una vez que se han encontrado modelos de genes estadísticamente significativos, los coeficientes de regresión de los modelos se pueden utilizar para identificar las condiciones para las cuales los genes muestran cambios de perfil estadísticamente significativos. Para ello, se obtiene un nuevo modelo sólo para genes seleccionados, aplicando una estrategia de selección de variables llamada *stepwise regression*.

El paquete maSigPro proporciona también una serie de funciones para el análisis visual de los resultados. Cuando el número de genes seleccionados es grande, se pueden aplicar algoritmos de clustering para dividir los datos en grupos de perfiles de expresión similares. Para ello se han incorporado una serie de algoritmos de clustering tradicionales.

# 5. Conceptos

## Valor de P

En contrastes de hipótesis y estadística general, el valor p también conocido como p-valor, se define como la probabilidad de obtener un resultado al menos tan extremo como el que se ha obtenido. Se rechaza la hipótesis nula si el valor p asociado al resultado observado es igual o menor que el nivel de significación establecido, convencionalmente 0,05 ó 0,01. Es decir, el valor p nos muestra la probabilidad de haber obtenido el resultado que hemos obtenido si suponemos que la hipótesis nula es cierta.
El valor p oscila entre 0 y 1: valores altos de p  no permiten rechazar la H0 y valores bajos de p rechazan la H0.

## Valor de P ajustado

El valor de p ajustado indica qué comparaciones de nivel de factor dentro de una familia de comparaciones (pruebas de hipótesis) son significativamente diferentes. Si el valor p ajustado es menor que alfa, entonces rechaza la hipótesis nula.

El ajuste limita la tasa de error por familia al nivel de significancia que se elija. Si se utiliza un valor p regular para comparaciones múltiples, la tasa de error por familia aumenta con cada comparación adicional.

## Fold change

Un fold change es una medida usada para comparar la expresión de genes entre dos conjuntos de muestras, por ejemplo, control y tratamiento.

Ejemplo de aplicación: en un experimento, en el control se tienen 50 lecturas y en el tratamiento se tienen 100 lecturas para el gen A. Esto significa que el gen A se expresa el doble en el tratamiento comparado con el control, o que el fold change es 2. Esto funciona bien para genes sobre expresados ya que el número corresponde directamente a cuántas veces un gen está sobre expresado. Pero cuando se trata de una sub expresión, el fold change será, por ejemplo, 0.5. De esta forma, todos los valores de genes sobreexpresados tendrán un valor de 1 a infinito, mientras que los valores de genes subexpresados estarán entre 0 y 1. Para nivelar esto, se utiliza el log2 para expresar el cambio de pliegue. En el ejemplo, log2 de 2 es 1, y log2 de 0.5 es -1.

## F-Estadística

Se utiliza con mayor frecuencia cuando se comparan modelos estadísticos que se han adaptado a un conjunto de datos, con el fin de identificar el modelo que mejor se adapte a la población de la que los datos fueron muestreados. Las "pruebas F" exactas surgen principalmente cuando los modelos se han ajustado a los datos utilizando mínimos cuadrados.

La hipótesis de que las medias de múltiples poblaciones normalmente distribuidas y con la misma desviación estándar son iguales es, quizás, la más conocida de las hipótesis verificada mediante el test F y el problema más simple del análisis de varianza.

## FDR (False Discovery Rate)

FDR es un método de conceptualización de la tasa de errores de tipo I (rechazos incorrectos de una verdadera hipótesis nula o falsos positivos) en la prueba de hipótesis nula cuando se realizan comparaciones múltiples. Los procedimientos de control de FDR están diseñados para controlar la proporción esperada de "descubrimientos" (hipótesis nulas rechazadas) que son falsos (rechazos incorrectos).

# 6. Implementación

## Primera fase: análisis de RNA-Seq. Confirmación de resultados.

### Obtención y preparación de las lecturas

Para realizar el análisis de expresión diferencial, se va a partir de los archivos .fastq, los cuales son los outputs de la secuenciación por RNA-Seq Illumina HiSeq 2000. Los archivos .fastq almacenan tanto la secuencia biológica como su correspondiente puntuación de calidad por cada base.

Para poder trabajar con este volumen de datos, es necesario usar el supercomputador Picasso. Para ello es necesaria la autorización previa, así como un usuario y contraseña. Una vez autorizado el acceso, se trabaja en el supercomputador desde el terminal de un ordenador local conectado a internet.

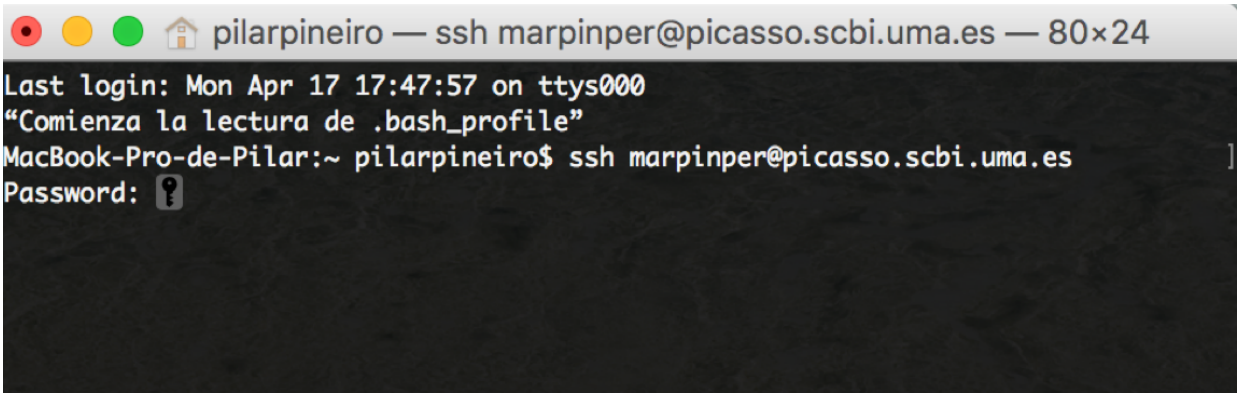



*Figura 3. Inicio de sesión en Picasso.*

En el escritorio de Picasso, cada usuario dispone de un escritorio dentro de su *unidad_z*, así como de un directorio llamado SCRATCH, al cual se puede acceder mediante el comando *cd $SCRATCH*, donde se dispone de mayor memoria. Este directorio no tiene backup, pero permite el almacén de un gran volumen de datos. Desde este directorio se operará con los archivos de las lecturas. Estas lecturas originales se encuentran en otro directorio de SCRATCH de Picasso, pero para poder manejarlas con mayor facilidad, se han renombrado las mismas creando un enlace a las originales en el directorio original.[3]

Para crear un enlace se utiliza el comando *ln -s origen destino.*

[3] Los nombres de los archivos originales y su correspondencia con el nombre otorgado para el mejor manejo de los archivos se incluyen en el documento Correspondencia.xslx.

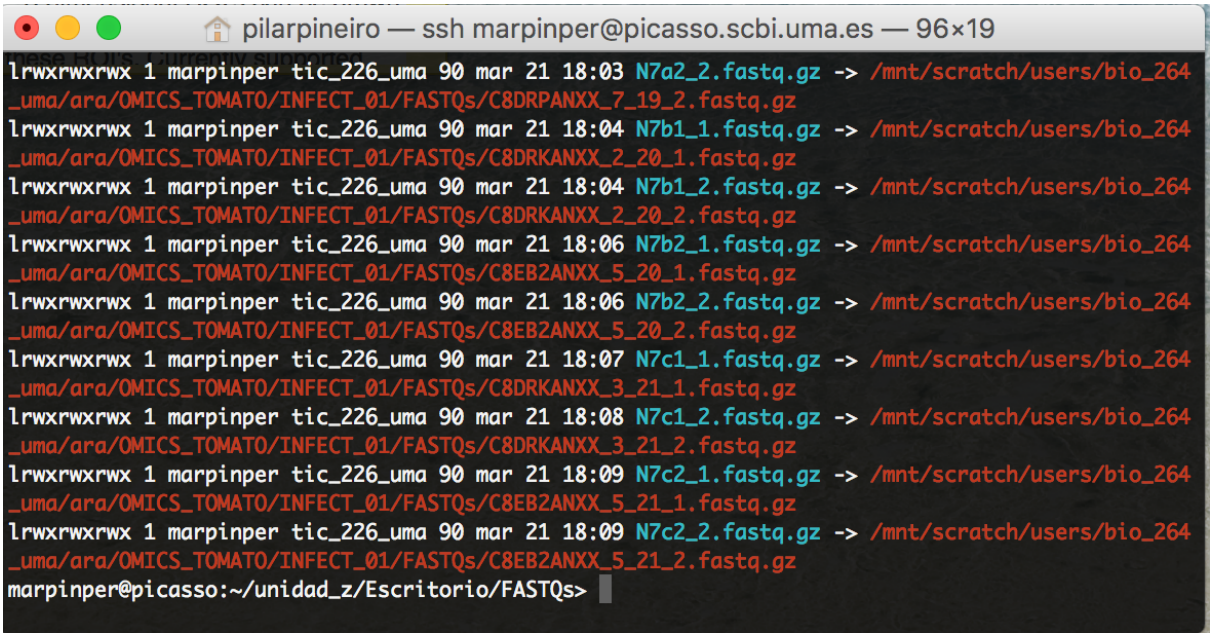



*Figura 4. Enlaces a los archivos .fastq originales.*

## Preprocesado

### Limpieza de las lecturas con SeqTrimNext

El preprocesado de las secuencias consiste en primer lugar en limpiar los archivos .fastq, y en este proyecto se utilizará SeqTrimNext. Para ello, se crea un directorio en SCRATCH llamado SEQTRIM donde se guardarán los outputs (los .fastq limpios).

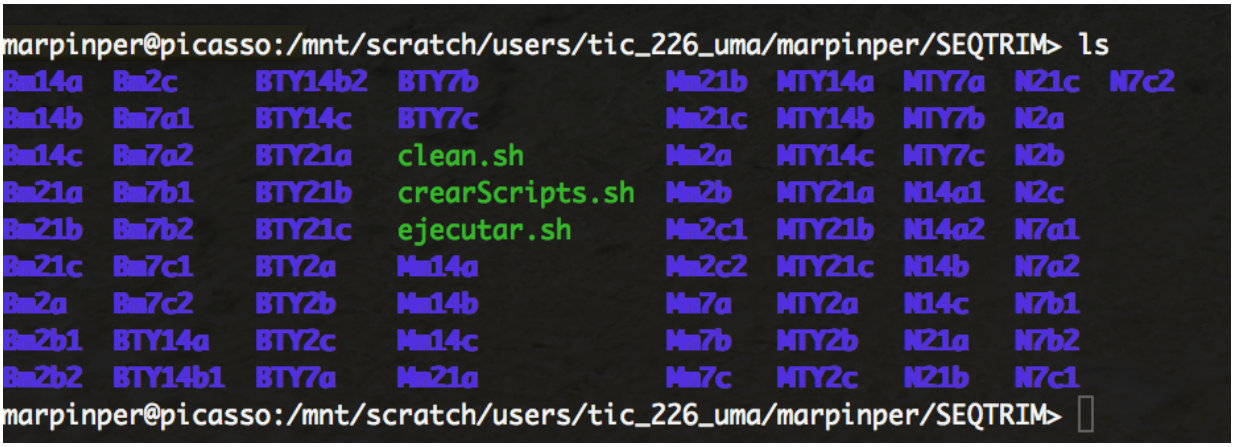



*Figura 5. Directorio SEQTRIM dentro de SCRATCH.*

Como se ha mencionado anteriormente, para ejecutar un programa en Picasso se debe desarrollar un script en el cual se cargue el programa a usar, se indique los

recursos necesarios para la ejecución y el comando con los parámetros pertinentes. Estos scripts se envían a cola con el comando *sbatch nombre_script.sh.*

En el comando de ejecución se debe indicar:

a. Plantilla .txt: Esta plantilla incluye las bases de datos de contaminantes y adaptadores necesarias según el tipo de secuencias que se deseen limpiar, los parámetros de tamaño de insert y calidad mínima. Se puede customizar una o utilizar una por defecto. En este caso se ha seleccionado la plantilla genomics_short_reads.tx, la cual se ajusta a las lecturas que se desean limpiar.

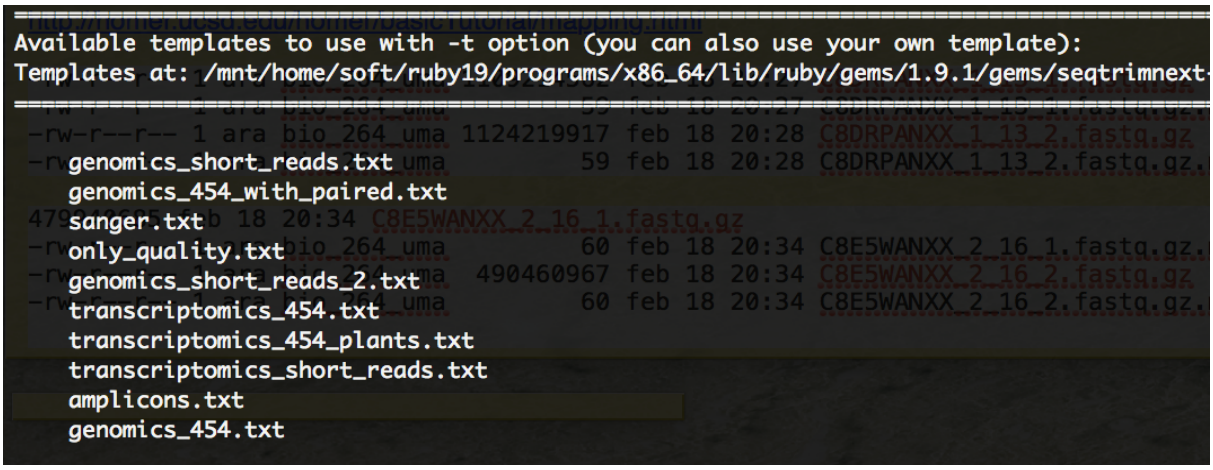



*Figura 6. Plantillas disponibles.*

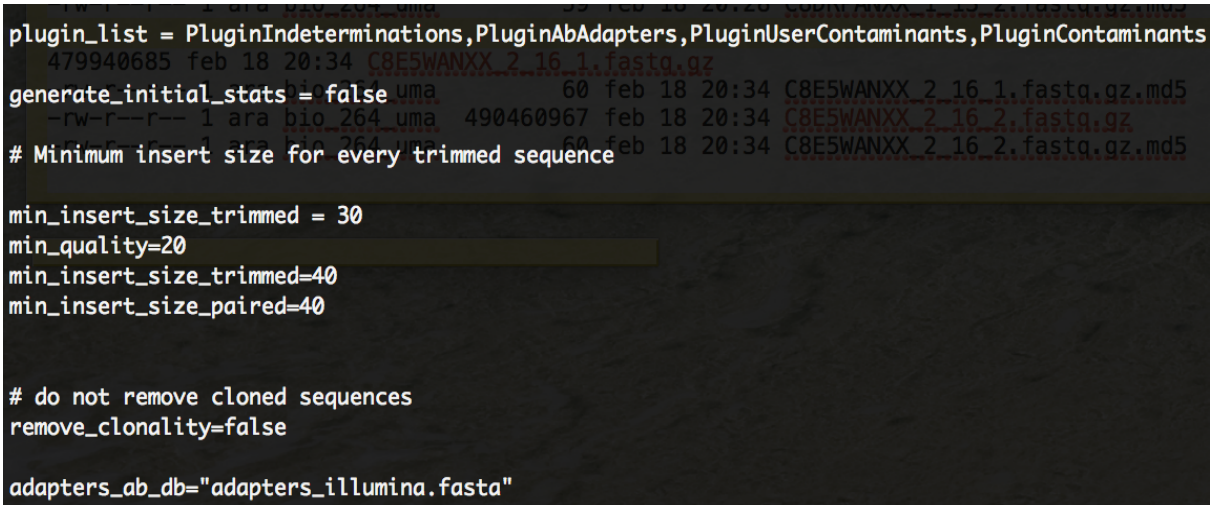



*Figura 7. Contenido de la plantilla genomics_short_reads.txt*

b. Archivos .fastq a limpiar. Estos corresponden a las lecturas pareadas y van separados por una coma.

c. Número de workers. Se han seleccionado 6 workers (6 nodos).

d. –K . No se escribirá cada secuencia en el output log. (Opción no verbose).

Para la preparación de la ejecución de SeqTrimNext se desarrollan tres scripts:

- clean.sh
  Este es un script modelo para ejecutar SeqTrimNext sobre todos los archivos .fastq. Se utiliza como base para generar los demás scripts.

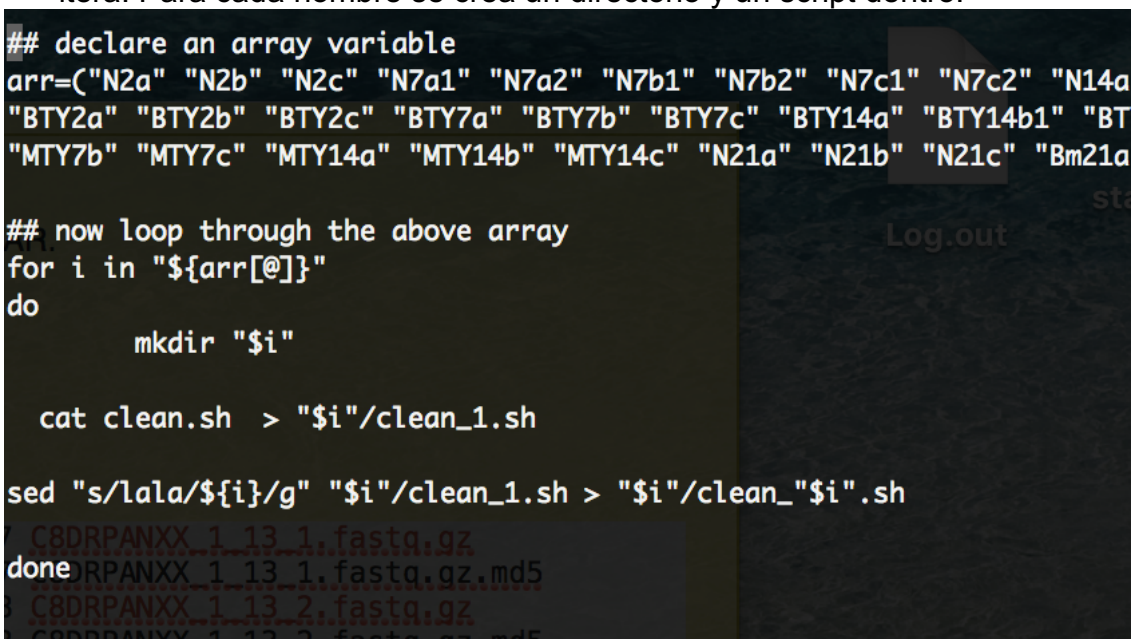
En este script se incluye el comando de ejecución con los parámetros y se indica que se necesitarán 6 nodos y 2gb de memoria, y que el tiempo de ejecución máximo permitido es de 7 días o 168 horas. Aunque se vayan a necesitar menos horas, poner un límite alto permitirá tener un margen de tiempo.

```
#!/usr/bin/env bash
# Leave only one comment symbol on selected options
# Those with two commets will be ignored:
# The name to show in queue lists for this job:
##SBATCH -J GENERATED_SCRIPT_FILE
# Number of desired cpus (can be in any node):
#SBATCH --ntasks=6

# Number of desired cpus (all in same node):
##SBATCH --cpus=1

# Amount of RAM needed for this job:
#SBATCH --mem=2gb
# The time the job will be running:
#SBATCH --time=168:00:00

# To use GPUs you have to request them:
##SBATCH --gres=gpu:1
# If you need nodes with special features uncomment the desired constraint line:
# * to request only the machines with 80 cores and 2TB of RAM
##SBATCH --constraint=bigmem
# * to request only machines with 16 cores and 64GB with InfiniBand network
##SBATCH --constraint=cal
# * to request only machines with 24 cores and Gigabit network
##SBATCH --constraint=slim

# Set output and error files
#SBATCH --error=job.%J.err
#SBATCH --output=job.%J.out

# MAKE AN ARRAY JOB, SLURM_ARRAYID will take values from 1 to 100
##SARRAY --range=1-100

# To load some software (you can show the list with 'module avail'):
module load seqtrimnext


# the program to execute with its parameters:
time seqtrimnext -t ~/unidad_z/Escritorio/genomics_short_reads.txt \
 -Q ~/unidad_z/Escritorio/FASTQs/lala_1.fastq.gz,~/unidad_z/Escritorio/FASTQs/lala_2.fastq.gz  -w 6 -K
```

*Figura 8. Script clean.sh*

- crearScripts.sh
  Este script utiliza el script clean.sh para generar una carpeta por cada par de .fastq (secuencias pareadas) y un script de ejecución como el de la figura 8 dentro de cada carpeta.

  El script consta de un array de nombres de todos los .fastq sobre el que se itera. Para cada nombre se crea un directorio y un script dentro.



```
## declare an array variable
arr=("N2a" "N2b" "N2c" "N7a1" "N7a2" "N7b1" "N7b2" "N7c1" "N7c2" "N14a
"BTY2a" "BTY2b" "BTY2c" "BTY7a" "BTY7b" "BTY7c" "BTY14a" "BTY14b1" "BT
"MTY7b" "MTY7c" "MTY14a" "MTY14b" "MTY14c" "N21a" "N21b" "N21c" "Bm21a

## now loop through the above array
for i in "${arr[@]}"
do
        mkdir "$i"

  cat clean.sh  > "$i"/clean_1.sh

sed "s/lala/${i}/g" "$i"/clean_1.sh > "$i"/clean_"$i".sh
done
```

*Figura 9. Script crearScripts.sh*

- ejecutar.sh
  Este script ejecuta el script correspondiente a cada muestra. De manera similar al script anterior, se recorre un array de nombres de los .fastq y se van ejecutando los scripts con *sbatch*.

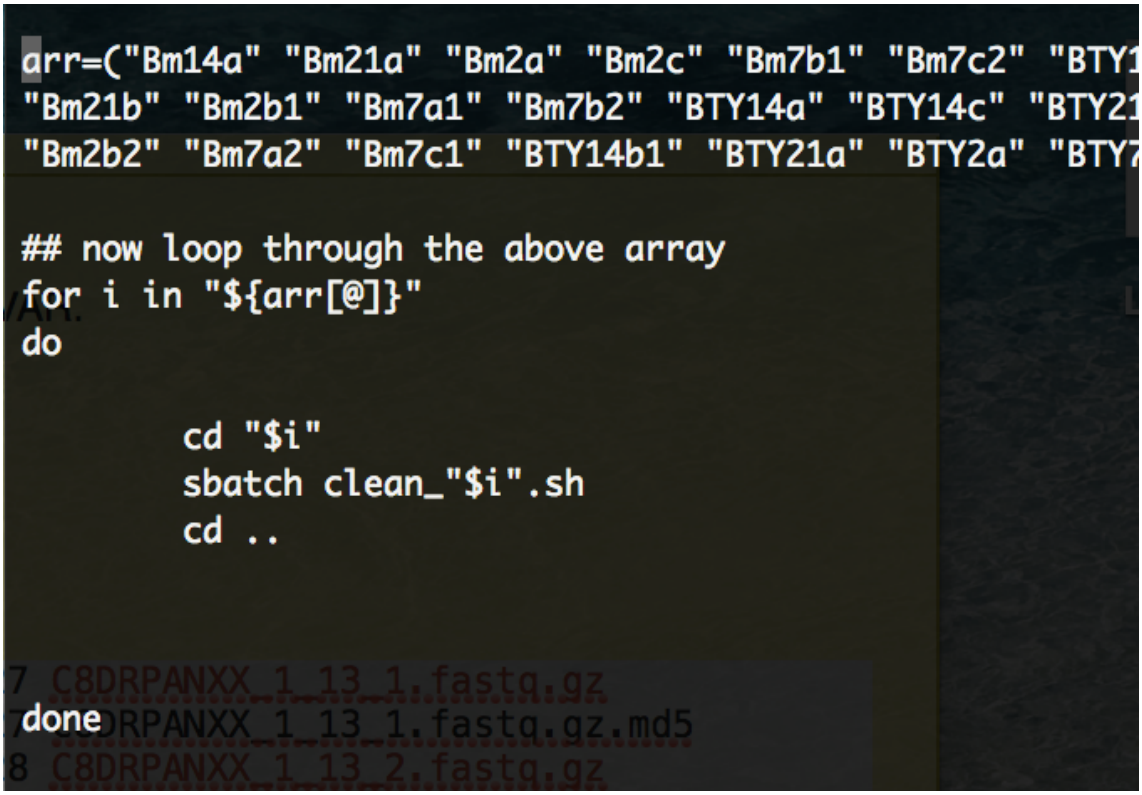



*Figura 10. Script ejecutar.sh*

Una vez se hayan enviado los scripts a cola, se puede supervisar su ejecución mediante el comando *squeue.* Si se requiere cancelar algún job, se debe ejecutar el comando *scancel numero_del_job.*

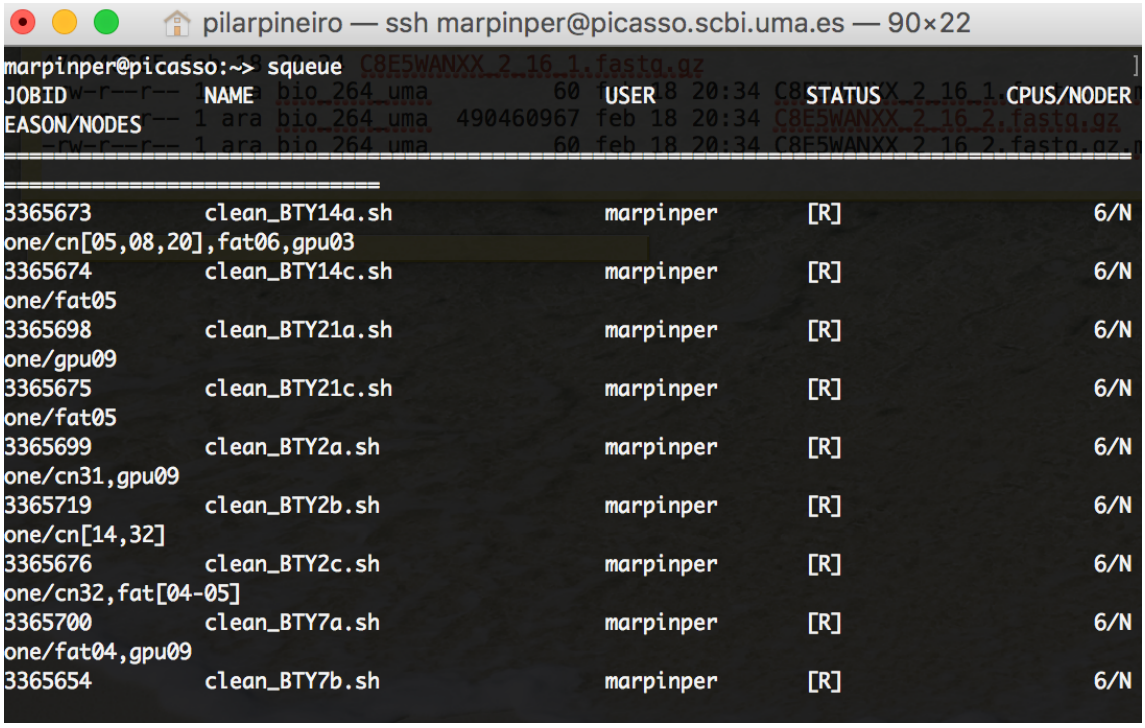



*Figura 11. Visualización de scripts en ejecución con el comando squeue.*

Una vez que se termina de ejecutar SeqTrimNext para una muestra, se generan diferentes outputs:

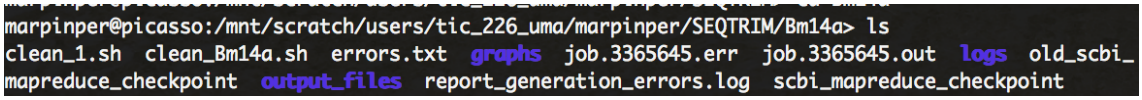



*Figura 12. Outputs generados por SeqTrimNext tras la ejecución.*

Dentro de la carpeta output_files se encuentran las lecturas limpias *paired_1_.fastq* y *paired_2_.fastq,* las cuales serán los inputs de la herramienta de mapeo STAR.

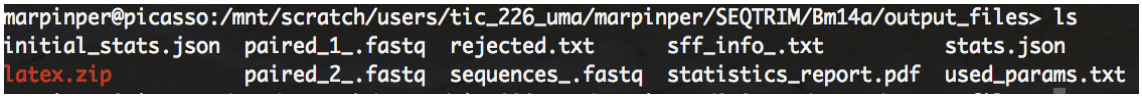



*Figura 13. Contenido de la carpeta output_files.*

Antes de comenzar a mapear, es necesario preprocesar los .fastq resultantes de la limpieza, ya que los nombres de las muestras no son compatibles con los nombres que espera recibir el mapeador. Para ello se desarrolló un script llamado procesa.sh. Para ejecutar este script es necesario cargar el módulo de SeqTrimNext:

*module load seqtrimnext*

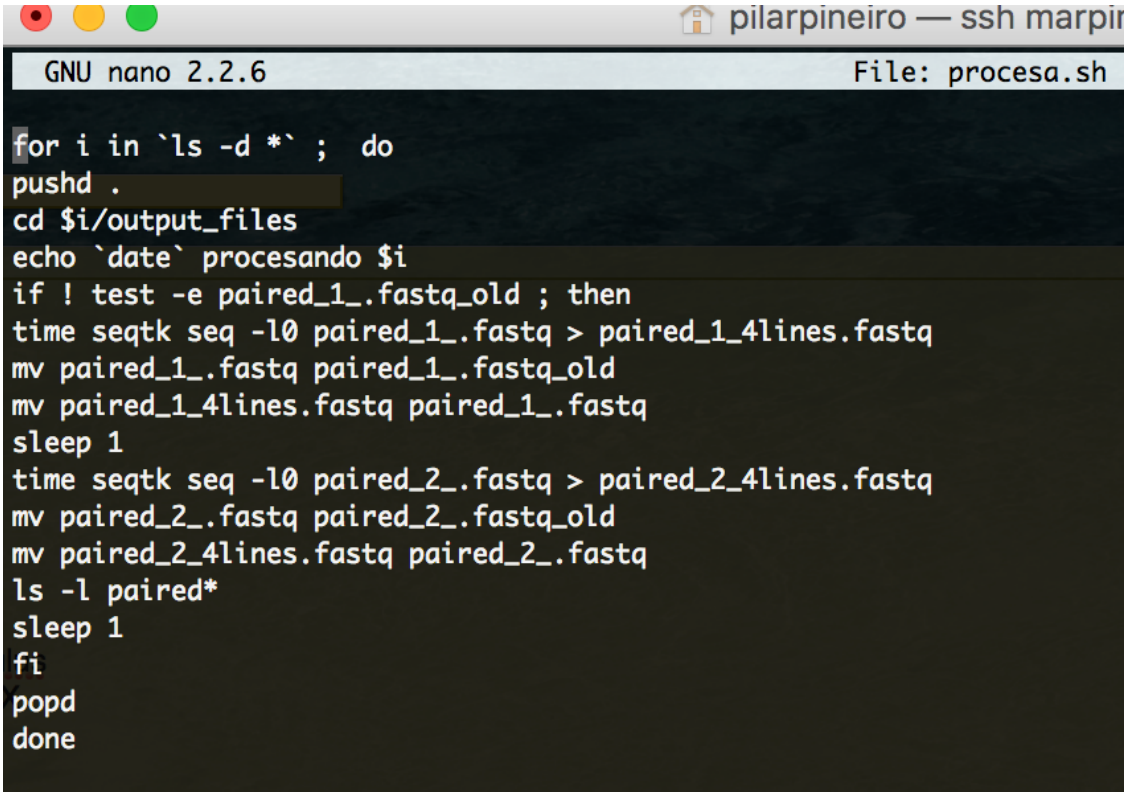



*Figura 14. Script procesa.sh que modifica los nombres de las muestras en cada lectura.*

Tras haber ejecutado este script se obtendrán cuatro archivos .fastq en cada carpeta de muestras:
*paired_1_.fastq, paired_1_.fastq_old, paired_2_.fastq, paired_2_.fastq_old*

Después de comprobar que se han procesado correctamente las lecturas, se deben borrar los antiguos .fastq (fastq_old) para liberar el espacio en memoria.

## Mapeo de las lecturas a un genoma de referencia con STAR

Tras haber limpiado las lecturas, se puede proceder a mapearlas al genoma de referencia.

Para mapear con STAR se debe crear primero un índice a partir de los archivos .fasta y .gtf (este último posee información acerca de la estructura de los genes) del genoma de referencia. El genoma de referencia utilizado es *Solanum_lycopersicum.SL2.50.31* de Ensembl. Es la versión 2.50 del release 31.

### *Creación del índice*

El primer paso para la creación del índice es descargar el genoma en formatos .gtf y .fasta de Ensembl. [20,21]

Para conectarse a Ensembl mediante ftp, se utilizan los comandos de la figura a continuación, utilizando como nombre de usuario anonymous y como contraseña una dirección de correo electrónico.

```
marpinper@picasso:/mnt/scratch/users/tic_226_uma/marpinper> ftp
ftp> open ftp.ensemblgenomes.org
Connected to ftp.ensemblgenomes.ebi.ac.uk.
220-
220- ftp.ensemblgenomes FTP server
220
Name (ftp.ensemblgenomes.org:marpinper): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
```

*Figura 15. Conexión a Ensembl.*

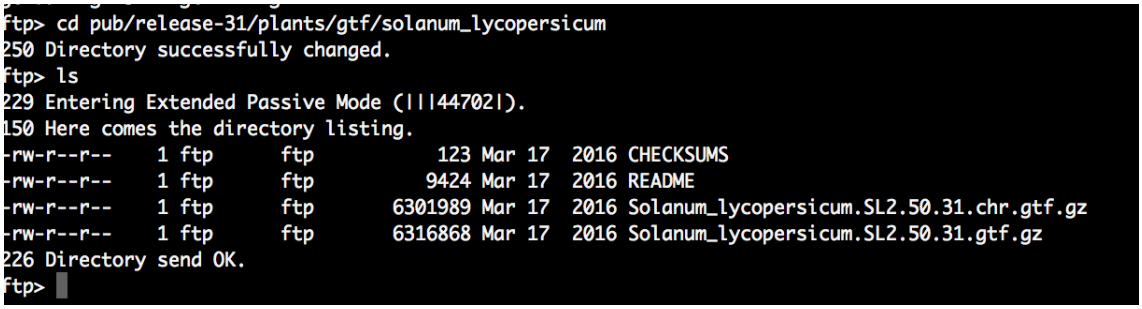



*Figura 16. Directorio gtf de Ensembl de Solanum_Lycopersicum.*

Se descarga el genoma Solanum_lycopersicum.SL2.50.31.gtf.gz con el comando *mget* y se repite el proceso para descargar el archivo en formato .fasta del genoma.

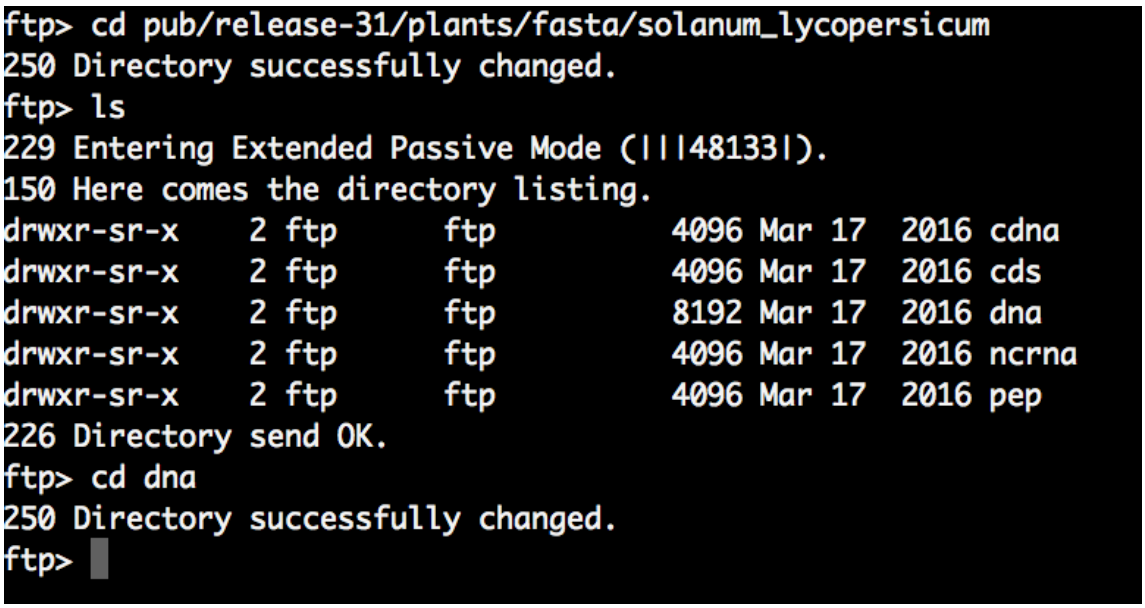



*Figura 17. Navegando por Ensembl  al directorio fasta.*

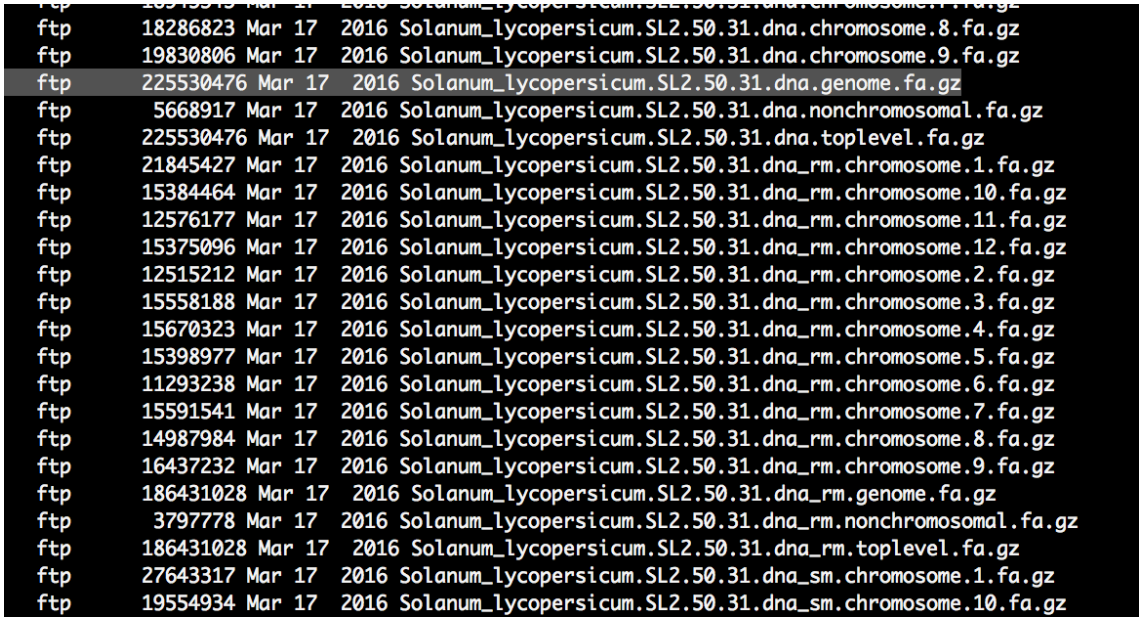



*Figura 18. Archivos .fasta en el directorio de Ensembl.*

Hay un gran número de archivos .fasta en el directorio de Ensembl. Se debe seleccionar el archivo del genoma completo, y éste no debe ser rm (repeat masked), ya que en este tipo de archivos se sustituyen las zonas repetitivas del genoma por 'N'.[22]

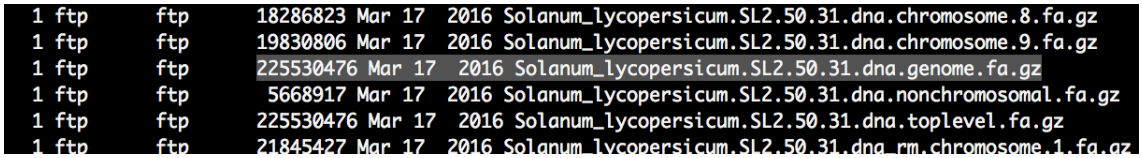



*Figura 19. Archivo a seleccionar en el directorio.*

Ambos archivos se deben almacenar en el mismo directorio. Para ello, se crea un directorio llamado *STAR* y dentro de él uno llamado *genome* y otro *mapear*. Ambos archivos se deben guardar en el directorio *genome*.

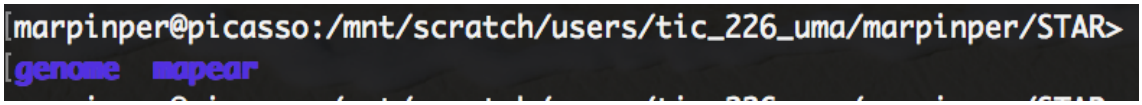
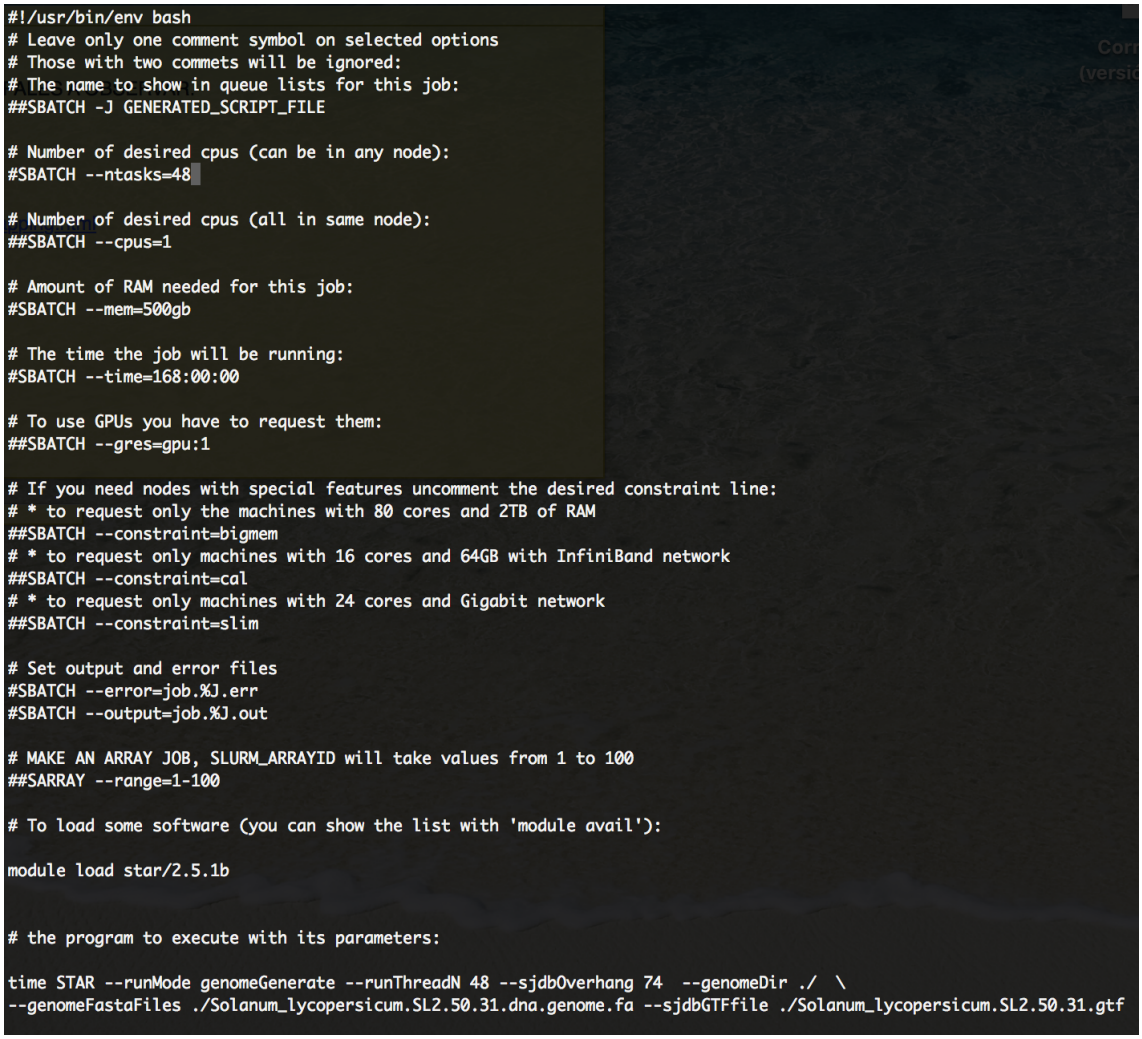



*Figura 20. Creación de directorios de trabajo para la creación del índice y el mapeo.*

Para la generación del índice es necesario desarrollar un script para enviar a cola. El script que se ha creado (index_create.sh) es el que se muestra a continuación.



```
#!/usr/bin/env bash
# Leave only one comment symbol on selected options
# Those with two commets will be ignored:
# The name to show in queue lists for this job:
##SBATCH -J GENERATED_SCRIPT_FILE

# Number of desired cpus (can be in any node):
#SBATCH --ntasks=48

# Number of desired cpus (all in same node):
##SBATCH --cpus=1

# Amount of RAM needed for this job:
#SBATCH --mem=500gb

# The time the job will be running:
#SBATCH --time=168:00:00

# To use GPUs you have to request them:
##SBATCH --gres=gpu:1

# If you need nodes with special features uncomment the desired constraint line:
# * to request only the machines with 80 cores and 2TB of RAM
##SBATCH --constraint=bigmem
# * to request only machines with 16 cores and 64GB with InfiniBand network
##SBATCH --constraint=cal
# * to request only machines with 24 cores and Gigabit network
##SBATCH --constraint=slim

# Set output and error files
#SBATCH --error=job.%J.err
#SBATCH --output=job.%J.out

# MAKE AN ARRAY JOB, SLURM_ARRAYID will take values from 1 to 100
##SARRAY --range=1-100

# To load some software (you can show the list with 'module avail'):

module load star/2.5.1b


# the program to execute with its parameters:

time STAR --runMode genomeGenerate --runThreadN 48 --sjdbOverhang 74  --genomeDir ./  \
--genomeFastaFiles ./Solanum_lycopersicum.SL2.50.31.dna.genome.fa --sjdbGTFfile ./Solanum_lycopersicum.SL2.50.31.gtf
```

*Figura 21. Script de generación de índice para mapear.*

En este script se piden los recursos necesarios (48 núcleos, 500 gb) según lo indicado en el manual de STAR [23].

El comando de STAR para generar el índice consta de:

a. –runMode genomeGenerate. Utilizado en STAR únicamente cuando se va a generar un índice.
b. –runThreadN 48. Indica el número de hebras.
c. –genomeDir. Indica dónde se encontrará el índice.
d. –genomeFastaFiles. Indica dónde se encuentra el genoma en formato .fasta.
e. – sjdbGTFfile. Indica dónde se encuentra el genoma en formato .gtf.
f. –sjdbOverhang 74. Longitud de las lecturas – 1.

Una vez generado el índice, el directorio *genome* contendrá los siguientes outputs:

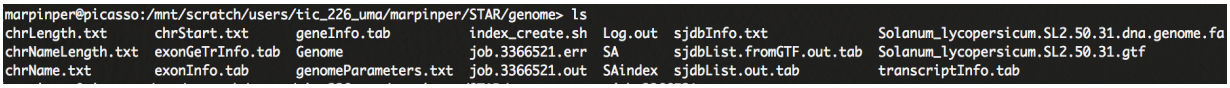



*Figura 22. Outputs de la ejecución de index_create.sh*

Cuando ejecutemos STAR para mapear indicaremos que el directorio que contiene el índice es *genome*. Este índice servirá para todos los mapeos que se realicen.

### *Mapeo y cuantificación*

De la misma forma que se ejecutó SeqTrimNext y se creó el índice, se necesita un script para enviar a cola y mapear.

STAR permite mapear y cuantificar a la vez con solo añadir un argumento más *[23*]. El comando de mapeo y cuantificación de STAR consta de:

a. --runThreadN. Número de hebras para la ejecución.
b. --runMode alingReads. Modo de ejecución de STAR.
c. --genomeDir. Ubicación del índice generado anteriormente.
d. --sjdbGTFfile. Ubicación del genoma en formato .gtf.
e. --sjdbOverhang. Longitud de las lecturas -1.
f. --readFilesIn. Archivos .fastq a mapear. Las secuencias paredas deben ir separadas por un espacio. Cuando se trata de dos archivos por cada pareada (archivos que se han dividido en dos porque el experimento se ha llevado a cabo en distintos lanes[4]), se deben separar por una coma “,”:

   ***MuestraA1_pareada1.fastq,MuestraA2_pareada1.fastq***
   ***MuestraA1_pareada2.fastq,MuestraA2_pareada2.fastq***

g. --outSAMtype BAM sortedByCoordinate. Tipo de archivo bam de salida. Este tipo de archivo es opcional, ya que se usará el recuento de genes de STAR.
h. --quantMode GeneCounts. Opción que cuenta el número de reads por gen. Un read o lectura es contado si solapa un y solo un gen. Ambos extremos de lecturas pareadas se tienen en cuenta para solapamiento. Los counts coinciden con los resultados de HTSeq-count. El output que se obtendrá es ReadsPerGene.out.tab con 4 columnas las cuales corresponden a diferentes opciones de ‘strand’ o hebra:

   - columna 1: gene ID
   - columna 2: counts para RNA-seq un-stranded
   - columna 3: counts para 1ra hebra alineada con RNA
   - columna 4: counts para 1ra hebra alineada con RNA (opción htseq-count -s reverse)

[4] Lane: carril físico en una celda de flujo que entra en la máquina de secuenciación.

En este proyecto se usará la columna 4 para generar las matrices de análisis de expresión diferencial.

Dentro del directorio *mapear* creado anteriormente (ver figura 20), se crearán los scripts para mapear cada .fastq.

```
#!/usr/bin/env bash
# Leave only one comment symbol on selected options
# Those with two commets will be ignored:
# The name to show in queue lists for this job:
##SBATCH -J GENERATED_SCRIPT_FILE

# Number of desired cpus (can be in any node):
#SBATCH --ntasks=48

# Number of desired cpus (all in same node):
##SBATCH --cpus=1

# Amount of RAM needed for this job:
#SBATCH --mem=500gb

# The time the job will be running:
#SBATCH --time=168:00:00

# To use GPUs you have to request them:
##SBATCH --gres=gpu:1

# If you need nodes with special features uncomment the desired constraint line:
# * to request only the machines with 80 cores and 2TB of RAM
##SBATCH --constraint=bigmem
# * to request only machines with 16 cores and 64GB with InfiniBand network
##SBATCH --constraint=cal
# * to request only machines with 24 cores and Gigabit network
##SBATCH --constraint=slim

# Set output and error files
#SBATCH --error=job.%J.err
#SBATCH --output=job.%J.out

# MAKE AN ARRAY JOB, SLURM_ARRAYID will take values from 1 to 100
##SARRAY --range=1-100
# To load some software (you can show the list with 'module avail'):

module load star/2.5.1b


# the program to execute with its parameters:

time STAR --runThreadN 48 --runMode alignReads --genomeDir /mnt/scratch/users/tic_226_uma/marpinper/STAR/genome/ \
--sjdbGTFfile /mnt/scratch/users/tic_226_uma/marpinper/STAR/genome/Solanum_lycopersicum.SL2.50.31.gtf \
--sjdbOverhang 74 --readFilesIn /mnt/scratch/users/tic_226_uma/marpinper/SEQTRIM/lala/output_files/paired_1_.fastq \
/mnt/scratch/users/tic_226_uma/marpinper/SEQTRIM/lala/output_files/paired_2_.fastq --outSAMtype BAM SortedByCoordinate --quantMode GeneCounts
```

*Figura 23. Script ejemplo de STAR. Los scripts de muestras divididas por diferentes lanes varían en la separación con una coma de dichas muestras.*

Usando un script modelo como este, se desarrollará un código que genere los directorios de mapeo y los scripts pertinentes.

```
arr=("N2a" "N2b" "N2c" "Bm14a" "N14b" "N14c" "Bm2a" "Bm2c" "Bm14a" "Bm14c" "BTY2a"
"Mm14a" "Mm14b" "Mm14c" "MTY2a" "MTY2b" "MTY2c" "MTY7a" "MTY7b" "MTY7c" "MTY14a" "M
"MTY21a" "MTY21b" "MTY21c")

# now loop through the above array
for i in "${arr[@]}"
do
        mkdir "$i"

  cat map_model.sh  > "$i"/map_model_1.sh

sed "s/lala/${i}/g" "$i"/map_model_1.sh > "$i"/map_model_"$i".sh

done
```

*Figura 24. Script de generación de scripts de mapeo.*

**NOTA: En el caso de las lecturas divididas en dos archivos por lanes, se crearon los scripts aparte.**

En la siguiente figura se muestra el resultado de la creación de los directorios y los scripts de mapeo.

```
marpinper@picasso:/mnt/scratch/users/tic_226_uma/marpinper/STAR/mapear> ls
Bm14a  Bm21c  Bm7b    BTY21a  BTY2c  Mm14b  Mm2a  Mm7c    MTY21b  MTY7a  N14c  N2b  scripts
Bm14b  Bm2a   Bm7c    BTY21b  BTY7a  Mm14c  Mm2b  MTY14a  MTY21c  MTY7b  N21a  N2c
Bm14c  Bm2b   BTY14a  BTY21c  BTY7b  Mm21a  Mm2c  MTY14b  MTY2a   MTY7c  N21b  N7a
Bm21a  Bm2c   BTY14b  BTY2a   BTY7c  Mm21b  Mm7a  MTY14c  MTY2b   N14a   N21c  N7b
Bm21b  Bm7a   BTY14c  BTY2b   Mm14a  Mm21c  Mm7b  MTY21a  MTY2c   N14b   N2a   N7c
```

*Figura 25 . Resultado de la creación de directorios para el mapeo.*

Para la ejecución, se procederá con *sbatch nombre del script*. Se puede generar un pequeño script que lo haga como se hizo anteriormente (ver figura 10).

A continuación se muestra un ejemplo de los archivos de salida obtenidos tras el mapeo.

```
marpinper@picasso:/mnt/scratch/users/tic_226_uma/marpinper/STAR/mapear/BTY14b> ls
Aligned.sortedByCoord.out.bam  job.3370094.out  Log.out           map_model_BTY14b.sh  ReadsPerGene.out.tab  _STARgenome
job.3370094.err                Log.final.out    Log.progress.out  map_model_d_1.sh     SJ.out.tab
marpinper@picasso:/mnt/scratch/users/tic_226_uma/marpinper/STAR/mapear/BTY14b>
```

*Figura 26 . Outputs de la ejecución de un mapeo.*

Una vez se ha terminado de mapear y cuantificar, se procede a elaborar la matriz de análisis para pasarle al script en R para realizar el análisis de expresión diferencial.

Es conveniente renombrar los archivos ReadsPerGene.out.tab con el nombre de la muestra de la forma: *nombremuestra_ReadsPerGene.out.tab.* Para ello, el script que se puede usar es el que se muestra a continuación.

```
arr=("Bm14a" "Bm14c" "Bm21a" "Bm21b" "Bm21c" "Bm2a" "Bm2b" "Bm2c" "B
"Mm14b" "Mm14c" "Mm21a" "Mm21b" "Mm21c" "Mm2b" "Mm2c" "Mm7a" "Mm7b"
"N21a" "N21b" "N21c" "N2a" "N2b" "N2c" "N7a" "N7b" "N7c")
for i in "${arr[@]}"
do
if [ -e "$i"/"ReadsPerGene.out.tab" ]; then
mv "$i"/ReadsPerGene.out.tab "$i"/"$i"_ReadsPerGene.out.tab
fi
done
```

*Figura 27. Script para la modificación de los nombres de los resultados.*

Para descargar las muestras, hay que conectarse a Picasso mediante conexión sftp. El procedimiento es similar a la conexión por ssh, y para descargar un archivo el comando a usar es *get nombrearchivo*.

```
pilarpineiro — ssh ◂ sftp marpinper@picasso.scbi.uma.es — 80
Last login: Wed May  3 13:59:00 on ttys001
"Comienza la lectura de .bash_profile"
MacBook-Pro-de-Pilar:~ pilarpineiro$ sftp marpinper@picasso.scbi.uma.es
Password:
Connected to picasso.scbi.uma.es.
sftp> cd /mnt/scratch/users/tic_226_uma/marpinper
sftp> cd STAR/mapear
sftp> ls
BTY14a          BTY14b          BTY14c          BTY21a          BTY21b
BTY21c          BTY2a           BTY2b           BTY2c           BTY7a
BTY7b           BTY7c           Bm14a           Bm14b           Bm14c
Bm21a           Bm21b           Bm21c           Bm2a            Bm2b
Bm2c            Bm7a            Bm7b            Bm7c            MTY14a
MTY14b          MTY14c          MTY21a          MTY21b          MTY21c
MTY2a           MTY2b           MTY2c           MTY7a           MTY7b
MTY7c           Mm14a           Mm14b           Mm14c           Mm21a
Mm21b           Mm21c           Mm2a            Mm2b            Mm2c
Mm7a            Mm7b            Mm7c            N14a            N14b
N14c            N21a            N21b            N21c            N2a
N2b             N2c             N7a             N7b             N7c
cambiar.sh      comprobar.sh    scripts
sftp> get */*_ReadsPerGene.out.tab
```

*Figura 28 . Descarga de resultados desde Picasso a un ordenador local.*

## Análisis

Para reproducir el análisis realizado en el estudio previo, se va a utilizar el lenguaje R y el entorno RStudio. Se utilizará la misma librería que se usó en el estudio anterior, DESeq2.

Se aplicará el mismo script de R con ligeras combinaciones a los distintos datos divididos según días después de la infección: 2, 7, 14 y 21. El código que se explicará detalladamente corresponde al análisis de los datos a los 2 días después de la infección.

En primer lugar, se importan las librerías necesarias.

```
library("DESeq2")
library("RColorBrewer")
library("pheatmap")
```

Es necesario configurar el directorio de trabajo.

```
dir<-("/Users/pilarpineiro/Google\ Drive/TFG/TOMATE/R")
setwd(dir)
```

Para importar los archivos correspondientes a los 2 dpi, es conveniente recoger todos estos archivos en una misma carpeta llamada *dpi_2*. A continuación se muestra el script usado para leer todos los archivos de la carpeta, introducirlos por columnas en una matriz y nombrar las columnas y filas.

Es importante proporcionar las matrices de conteo como entrada para el modelo estadístico de DESeq2, ya que sólo los valores de recuento permiten evaluar la precisión de la medición correctamente. El modelo DESeq2 corrige internamente el tamaño de la biblioteca, por lo que los valores transformados o normalizados no deben utilizarse como entrada [24].

```
ff <- list.files( path = "/Users/pilarpineiro/Google\ Drive/TFG/TOMATE/RESULTS/dpi_2",
                  pattern = "_ReadsPerGene.out.tab$", full.names = TRUE )
counts.files <- lapply( ff, read.table, skip = 4 )
counts <- as.data.frame( sapply( counts.files, function(x) x[ , 4 ] ) )

ff <- gsub( "[.]ReadsPerGene[.]out[.]tab", "", ff )

ff <- gsub( "[.]/counts/", "", ff )

colnames(counts) <- c("Bm2a","Bm2b","Bm2c","BTY2a","BTY2b","BTY2c","Mm2a",
                      "Mm2b","Mm2c","MTY2a","MTY2b","MTY2c","N2a","N2b","N2c")

row.names(counts) <- counts.files[[1]]$V1
cts<- as.matrix(counts)
```

Una vez se tiene la matriz de cuentas, se puede comenzar a analizar los datos.

| | Bm2a | Bm2b | Bm2c | BTY2a | BTY2b | BTY2c | Mm2a | Mm2b | Mm2c | MTY2a | MTY2b | MTY2c | N2a | N2b | N2c |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NCRNA_1_256 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| NCRNA_1_241 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Solyc01g005000.2 | 2042 | 1601 | 1840 | 1683 | 1636 | 1231 | 2156 | 2002 | 1649 | 1266 | 1675 | 2604 | 1473 | 2427 | 1082 |
| Solyc01g005010.2 | 450 | 421 | 415 | 449 | 585 | 516 | 407 | 347 | 373 | 309 | 439 | 693 | 537 | 1582 | 611 |
| Solyc01g005020.2 | 1216 | 1324 | 1223 | 1191 | 1144 | 1272 | 1684 | 1570 | 1578 | 1221 | 1474 | 2131 | 1100 | 2269 | 1054 |
| Solyc01g005030.2 | 661 | 764 | 584 | 647 | 573 | 587 | 617 | 498 | 487 | 450 | 538 | 744 | 570 | 1033 | 427 |
| Solyc01g005040.2 | 7 | 10 | 5 | 7 | 10 | 4 | 17 | 18 | 11 | 9 | 8 | 12 | 5 | 17 | 15 |
| Solyc01g005050.2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Solyc01g005060.2 | 83 | 79 | 69 | 102 | 117 | 79 | 124 | 118 | 125 | 109 | 116 | 192 | 73 | 223 | 88 |
| Solyc01g005070.2 | 0 | 0 | 0 | 2 | 2 | 1 | 2 | 1 | 0 | 1 | 2 | 1 | 0 | 2 | 0 |
| Solyc01g005080.2 | 1651 | 1645 | 1623 | 1514 | 1560 | 1523 | 1063 | 952 | 939 | 797 | 861 | 1452 | 1735 | 3050 | 1433 |
| Solyc01g005090.2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| Solyc01g005100.2 | 406 | 517 | 387 | 409 | 389 | 415 | 506 | 500 | 457 | 415 | 468 | 734 | 367 | 755 | 387 |

*Figura 29. Primeras 13 filas de la matriz de cuentas.*

Para comenzar a analizar los datos se debe crear una tabla llamada *sampleTable*, la cual indica el diseño del experimento: número de réplicas y nombre de la condición.

```
directory <-("/Users/pilarpineiro/Google\ Drive/TFG/TOMATE/RESULTS/dpi_2")
sampleFiles <- c("N2a_ReadsPerGene.out.tab","N2b_ReadsPerGene.out.tab","N2c_ReadsPerGene.out.tab",
                 "Bm2a_ReadsPerGene.out.tab","Bm2b_ReadsPerGene.out.tab","Bm2c_ReadsPerGene.out.tab",
                 "BTY2a_ReadsPerGene.out.tab","BTY2b_ReadsPerGene.out.tab","BTY2c_ReadsPerGene.out.tab",
                 "Mm2a_ReadsPerGene.out.tab","Mm2b_ReadsPerGene.out.tab","Mm2c_ReadsPerGene.out.tab",
                 "MTY2a_ReadsPerGene.out.tab","MTY2b_ReadsPerGene.out.tab","MTY2c_ReadsPerGene.out.tab")

sampleCondition <- c("N2","N2","N2","Bm2","Bm2","Bm2","BTY2","BTY2","BTY2","Mm2","Mm2","Mm2","MTY2","MTY2","MTY2")

sampleTable<- data.frame(row.names=c("Bm2a","Bm2b","Bm2c","BTY2a","BTY2b","BTY2c","Mm2a","Mm2b","Mm2c",
                                     "MTY2a","MTY2b","MTY2c","N2a","N2b","N2c"), condition=as.factor(c(rep("Bm2",3),
                                        rep("BTY2", 3), rep("Mm2", 3), rep("MTY2", 3),rep("N2", 3))))
```

| | condition |
|---|---|
| Bm2a | Bm2 |
| Bm2b | Bm2 |
| Bm2c | Bm2 |
| BTY2a | BTY2 |
| BTY2b | BTY2 |
| BTY2c | BTY2 |
| Mm2a | Mm2 |
| Mm2b | Mm2 |
| Mm2c | Mm2 |
| MTY2a | MTY2 |
| MTY2b | MTY2 |
| MTY2c | MTY2 |
| N2a | N2 |
| N2b | N2 |
| N2c | N2 |

*Figura 30. Vista de sampleTable.*

Una vez se tiene la matriz de cuentas y la sampleTable, ambas se pasan como parámetro a la función *DESeqDataSetFromMatrix*, la cual construye el dataset *dds_2*. Se indica en *design* la columna condition de *sampleTable*.

En el prefiltrado se filtran los genes que tienen 0 o 1 lecturas. De esta manera se reduce el tamaño de memoria del dataset *dds_2* y se aumenta así la velocidad de las funciones de transformación y testeo [24].

```
dds_2 <- DESeqDataSetFromMatrix(countData = cts,colData = sampleTable,design = ~ condition)

##Pre-filtering
dds_2 <- dds_2[ rowSums(counts(dds_2)) > 1, ]
```

Los pasos estándar del análisis de expresión diferencial están empaquetados en una única función, DESeq.

```
dds_2 <- DESeq(dds_2)
```

Utilizando la función *results* se puede indicar entre qué condiciones se desea hacer un contraste de expresión de genes.

```
Comp_1<-results(dds_2, contrast=c("condition","N2","Bm2"))
Comp_2<-results(dds_2, contrast=c("condition","N2","BTY2"))
Comp_3<-results(dds_2, contrast=c("condition","N2","Mm2"))
Comp_4<-results(dds_2, contrast=c("condition","N2","MTY2"))
Comp_5<-results(dds_2, contrast=c("condition","Bm2","BTY2"))
Comp_6<-results(dds_2, contrast=c("condition","MTY2","BTY2"))
Comp_7<-results(dds_2, contrast=c("condition","MTY2","Mm2"))
```

Para cada comparación (Comp_*), se seleccionan los genes cuyo p-ajustado sea menor a 0.05: estos serán los genes significativos. Posteriormente, se ordenan y se cuenta el total de genes expresados diferencialmente.

```
Comp_1_resSig <- Comp_1[which(Comp_1$padj <0.05),]
head(Comp_1_resSig[order(Comp_1_resSig$log2FoldChange, decreasing = TRUE),])
nrow(Comp_1_resSig)
```

Este proceso se repite para cada punto temporal (dpi) y se recogen los resultados obtenidos para todos los puntos temporales. (Ver tablas 2, 3, 4, 5).

Para visualizar las muestras en mapas de calor[5], se aplicará la función *dist()* a la transformada de la matriz de cuentas traspuesta para obtener las distancias de muestra-a-muestra. Rlog o *Regularized logarithm* es una función que transforma los datos normalizándolos con respecto al tamaño de la librería.

[5] Un mapa de calor de la matriz de distancias proporciona una visión de las similitudes y diferencias entre muestras.

```
rld_2 <- rlog(dds_2, blind=F)

sampleDists <- dist(t(assay(rld_2)))

sampleDistMatrix <- as.matrix(sampleDists)
rownames(sampleDistMatrix) <- paste(rld_2$condition, rld_2$type, sep="-")
colnames(sampleDistMatrix) <- NULL
colors <- colorRampPalette( rev(brewer.pal(9, "Blues")) )(255)
pheatmap(sampleDistMatrix,
         clustering_distance_rows=sampleDists,
         clustering_distance_cols=sampleDists,
         col=colors)
```

## Resultados obtenidos: genes diferencialmente expresados

| | dpi2 | | | | |
|---|---|---|---|---|---|
| Conditions | Naive | Agro-mock | Agro+TYL | Mosca-mock | Mosca+TYL |
| Naive | | | | | |
| Agro-mock | 1910 | | | | |
| Agro+TYL | 1148 | 212 | | | |
| Mosca-mock | 9025 | | | | |
| Mosca+TYL | 9740 | | 9532 | 1133 | |

*Tabla 2. Resultados obtenidos para las muestras pasados 2 dpi.*

| | dpi7 | | | | |
|---|---|---|---|---|---|
| Conditions | Naive | Agro-mock | Agro+TYL | Mosca-mock | Mosca+TYL |
| Naive | | | | | |
| Agro-mock | 1854 | | | | |
| Agro+TYL | 5715 | 2104 | | | |
| Mosca-mock | 5213 | | | | |
| Mosca+TYL | 4843 | | 4450 | 61 | |

*Tabla 3. Resultados obtenidos para las muestras pasados 7 dpi.*

| | dpi14 | | | | |
|---|---|---|---|---|---|
| Conditions | Naive | Agro-mock | Agro+TYL | Mosca-mock | Mosca+TYL |
| Naive | | | | | |
| Agro-mock | 580 | | | | |
| Agro+TYL | 6888 | 5232 | | | |
| Mosca-mock | 14 | | | | |
| Mosca+TYL | 5822 | | 1610 | 5584 | |

*Tabla 4. Resultados obtenidos para las muestras pasados 14 dpi.*

| | dpi21 | | | | |
|---|---|---|---|---|---|
| Conditions | Naive | Agro-mock | Agro+TYL | Mosca-mock | Mosca+TYL |
| Naive | | | | | |
| Agro-mock | 294 | | | | |
| Agro+TYL | 8266 | 8615 | | | |
| Mosca-mock | 3096 | | | | |
| Mosca+TYL | 9194 | | 1883 | 10813 | |

*Tabla 5. Resultados obtenidos para las muestras pasados 21 dpi.*

En la tabla 6 se recoge un resumen de las tablas 2, 3, 4 y 5 de la misma forma que se hizo con los resultados originales, que se muestran en la tabla 7.

| ***DEGs_padj <0,05*** | ***2dpi*** | ***7dpi*** | ***14 dpi*** | ***21dpi*** |
|---|---|---|---|---|
| Agro_TYLCV/Agro_mock | 212 | 2104 | 5232 | 8615 |
| Bemisia_TYLCV/Bemisia_mock | 1133 | 61 | 5584 | 10813 |
| Agro_mock/Naive | 1910 | 1854 | 580 | 294 |
| Bemisia_mock/Naive | 9025 | 5213 | 14 | 3096 |

*Tabla 6. Resultados obtenidos con DESeq2.*

| ***DEGs_padj <0,05*** | ***2dpi*** | ***7dpi*** | ***14 dpi*** | ***21dpi*** |
|---|---|---|---|---|
| Agro_TYLCV/Agro_mock | 321 | 1909 | 3754 | 8417 |
| Bemisia_TYLCV/Bemisia | 1006 | 76 | 4171 | 10422 |
| Agro_mock/Naive | 1970 | 1746 | 270 | 298 |
| Bemisia/Naive | 9181 | 5465 | 0 | 2944 |

*Tabla 7. Resultados del estudio realizado por la empresa CNAG.*

En esta comparación se puede ver que los resultados de ambas tablas siguen la misma tendencia.

## Resultados obtenidos: mapas de calor

Los mapas de calor de las figuras 31, 32, 33 y 34 proporcionan una visión general de las similitudes y diferencias entre muestras [24].

La matriz de cuentas de lecturas a los 2 dpi (ver figura 31) refleja la diferenciación de los genes de las muestras al transcurrir 2 días: existe una diferenciación relacionada con la presencia de mosca y bacteria: se observa que a los 2 días la planta no reacciona a la presencia o ausencia de infección.

La matriz de cuentas de lecturas a los 7 dpi (ver figura 32) refleja la diferenciación de los genes al transcurrir 7 días: los genes se agrupan según se trate de plantas infectadas con bacteria y plantas infectadas con mosca. A su vez las plantas expuestas a bacteria y mosca sin infectar se agrupan en función del tipo de exposición:

mosca o bacteria a excepción de una de las muestras expuesta a bacteria (Bm7), que se agrupa con las muestras infectadas. Los genes de las plantas 'naive', sin tratar, se agrupan por otro lado.

Las matrices de cuentas de lecturas a los 14 y 21 dpi (ver figuras 33 y 34) reflejan la diferenciación de los genes al transcurrir 14 y 21 días desde la infección: los genes de las plantas infectadas se agrupan entre sí al igual que los de las plantas no infectadas, independientemente de si el vector de transmisión es mosca o bacteria.

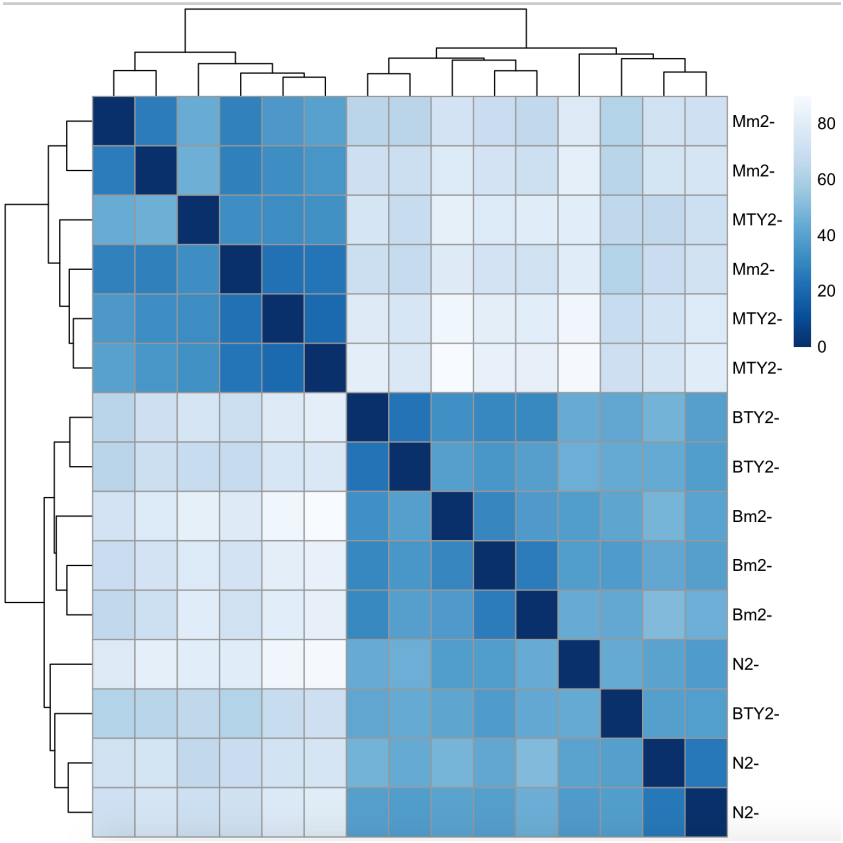



*Figura 31. Mapa de calor de las lecturas a los 2 dpi.*

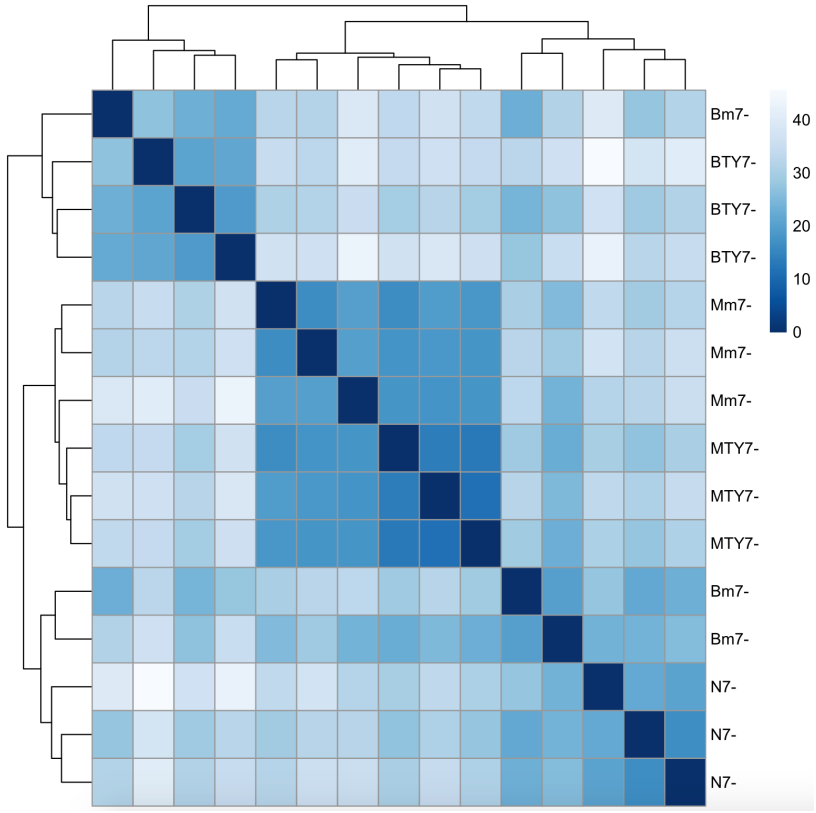



*Figura 32. Mapa de calor de las lecturas a los 7 dpi*

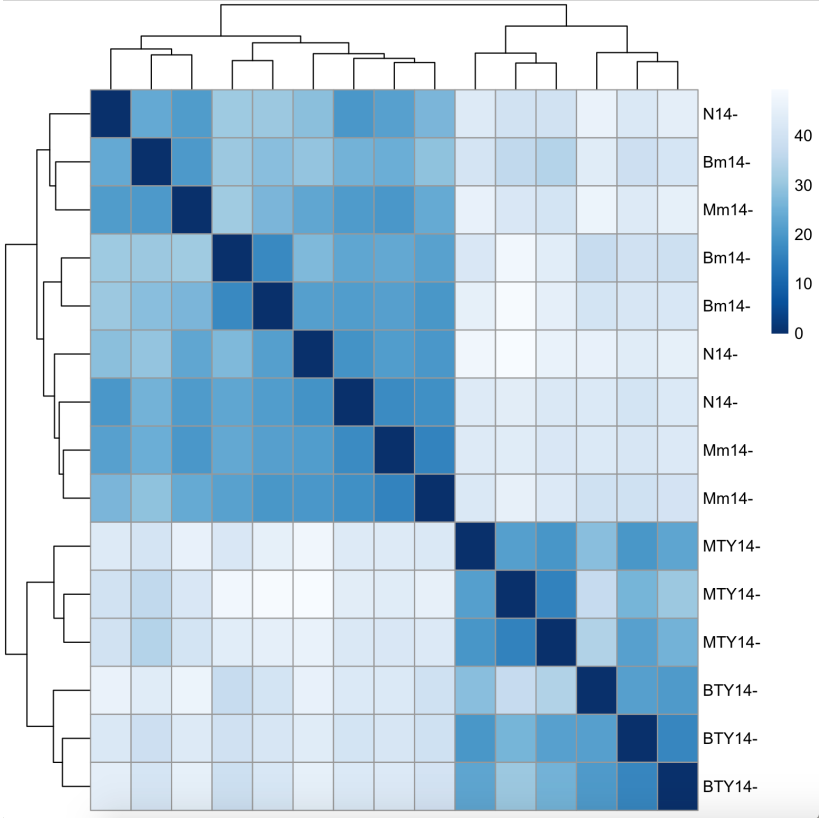



*Figura 33. Mapa de calor de las lecturas a los 14 dpi.*

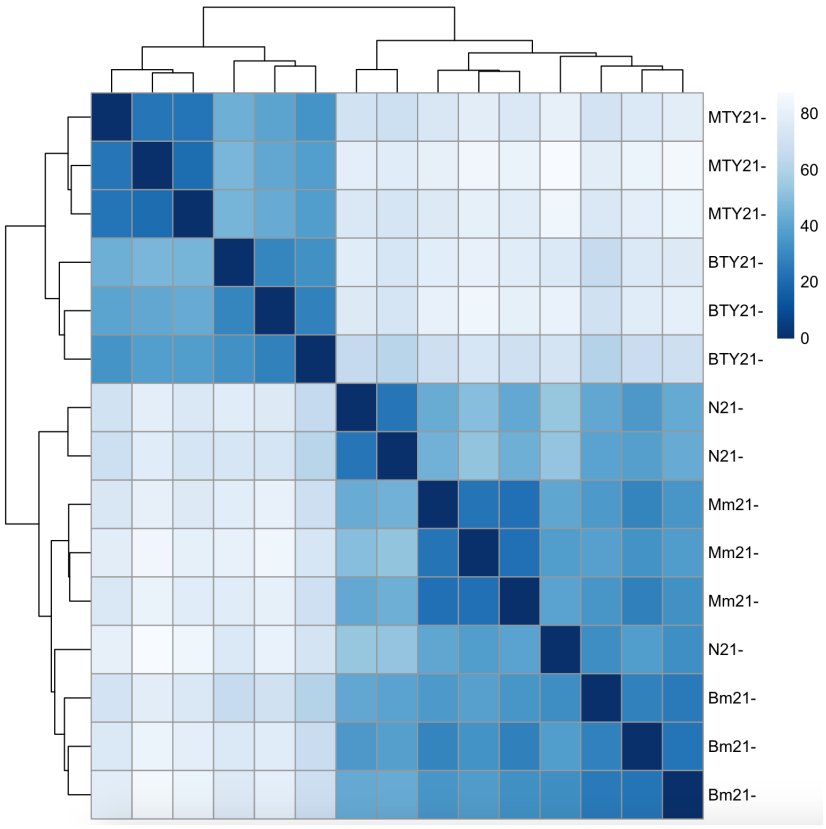



*Figura 34. Mapa de calor de las lecturas a los 21 dpi.*

Los resultados indican la tendencia de los genes en función de las condiciones y el tiempo transcurrido. Se puede observar qué genes que se expresan de forma diferente según la condición bajo la que se encuentran. Pero, ¿Se puede saber de esta forma qué genes se corregulan?¿Son los mismos genes los que se desregulan en presencia de mosca? Para responder a ello se necesita un análisis más complejo.

## Segunda fase: Análisis temporal de expresión diferencial

Tras haber comprobado que los resultados obtenidos coinciden con los del estudio previo, se va a proceder a la realización de un estudio más amplio. Con la librería de R *maSigPro* se va a hacer un análisis de todas las muestras en conjunto.

En primer lugar, se importan las librerías que se van a utilizar.

```
library(edgeR)
library(maSigPro)
library(MASS)
```

Para importar las lecturas, se sigue el mismo procedimiento para cada carpeta: *dpi_14, dpi_21, dpi_2, dpi_7.* Este procedimiento es el mismo que se ha usado para el análisis de expresión diferencial con DESeq2. En las figuras a continuación se muestra la creación de la matriz de lecturas a los 14 dpi.

```
ff <- list.files( path = "/Users/pilarpineiro/Google\ Drive/TFG/TOMATE/RESULTS/dpi_14",
                  pattern = "_ReadsPerGene.out.tab$",
                  full.names = TRUE )
counts.files <- lapply( ff, read.table, skip = 4 )
counts <- as.data.frame( sapply( counts.files, function(x) x[ , 4 ] ) )
ff <- gsub( "[.]ReadsPerGene[.]out[.]tab", "", ff )
ff <- gsub( "[.]/counts/", "", ff )
colnames(counts) <- c("Bm14a","Bm14b","Bm14c","BTY14a","BTY14b",
                      "BTY14c","Mm14a","Mm14b","Mm14c","MTY14a",
                      "MTY14b","MTY14c","N14a","N14b","N14c")
row.names(counts) <- counts.files[[1]]$V1
cts_14<- as.matrix(counts)
```

Una vez se tienen todas las matrices de cuentas, se normalizan. En la figura a continuación se continua con el ejemplo de 14_dpi.

```
expr_14<- DGEList(counts=cts_14,group=colnames(cts_14))
expr_14<- calcNormFactors(expr_14)
ncounts_14<-cpm(expr_14, normalized.lib.sizes=TRUE)
```

A partir de las matrices de cuentas normalizadas, se forma una única matriz, la cual se va a analizar.

```
matrix_counts<-cbind(ncounts_14,ncounts_2,ncounts_21,ncounts_7)
```

Es necesario crear una matriz de diseño, en la cual se especifica el número de réplicas para cada condición así como el tiempo.

Se crea una matriz de 60 filas (60 condiciones) y 7 columnas (estas columnas contienen la información sobre el tiempo, número de réplicas, y el tipo de condición).

```
matrix_design<-matrix(nrow=60,ncol=7)

rownames(matrix_design)<-as.character(colnames(matrix_counts))

colnames(matrix_design)<-c("Time","Replicate","Control","Mockmosca","Mockbact","InfMosca","InfBact")

matrix_design[,1]<-c(14,14,14,14,14,14,14,14,14,14,14,14,14,14,14,2,2,2,2,2,2,2,2,2,2,2,2,2,
                     21,21,21,21,21,21,21,21,21,21,21,21,21,21,7,7,7,7,7,7,7,7,7,7,7,7,7,7)
matrix_design[,2]<-c(1,1,1,2,2,2,3,3,3,4,4,4,5,5,5,6,6,6,7,7,7,8,8,8,9,9,9,10,10,10,11,11,11,12,
                     12,12,13,13,13,14,14,14,15,15,15,16,16,16,17,17,17,18,18,18,19,19,19,20,20,20)
matrix_design[,3]<-c(0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,
                     0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1)
matrix_design[,4]<-c(0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,
                     1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0)
matrix_design[,5]<-c(1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,
                     0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0)
matrix_design[,6]<-c(0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,
                     0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0)
matrix_design[,7]<-c(0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,
                     0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0)
```

En la figura siguiente se muestra la matriz de diseño para el experimento.

| | Time | Replicate | Control | Mockmosca | Mockbact | InfMosca | InfBact |
|---|---|---|---|---|---|---|---|
| Bm14a | 14 | 1 | 0 | 0 | 1 | 0 | 0 |
| Bm14b | 14 | 1 | 0 | 0 | 1 | 0 | 0 |
| Bm14c | 14 | 1 | 0 | 0 | 1 | 0 | 0 |
| BTY14a | 14 | 2 | 0 | 0 | 0 | 0 | 1 |
| BTY14b | 14 | 2 | 0 | 0 | 0 | 0 | 1 |
| BTY14c | 14 | 2 | 0 | 0 | 0 | 0 | 1 |
| Mm14a | 14 | 3 | 0 | 1 | 0 | 0 | 0 |
| Mm14b | 14 | 3 | 0 | 1 | 0 | 0 | 0 |
| Mm14c | 14 | 3 | 0 | 1 | 0 | 0 | 0 |
| MTY14a | 14 | 4 | 0 | 0 | 0 | 1 | 0 |
| MTY14b | 14 | 4 | 0 | 0 | 0 | 1 | 0 |
| MTY14c | 14 | 4 | 0 | 0 | 0 | 1 | 0 |
| N14a | 14 | 5 | 1 | 0 | 0 | 0 | 0 |
| N14b | 14 | 5 | 1 | 0 | 0 | 0 | 0 |
| N14c | 14 | 5 | 1 | 0 | 0 | 0 | 0 |
| Bm2a | 2 | 6 | 0 | 0 | 1 | 0 | 0 |
| Bm2b | 2 | 6 | 0 | 0 | 1 | 0 | 0 |
| Bm2c | 2 | 6 | 0 | 0 | 1 | 0 | 0 |
| BTY2a | 2 | 7 | 0 | 0 | 0 | 0 | 1 |
| BTY2b | 2 | 7 | 0 | 0 | 0 | 0 | 1 |
| BTY2c | 2 | 7 | 0 | 0 | 0 | 0 | 1 |

*Figura 35. Matriz de diseño. Primeras 21 filas de 60.*

La función *make.design.matrix()* crea una matriz de regresión para el modelo de regresión completo.

```
design <- make.design.matrix(matrix_design)
```

Se calcula un ajuste de regresión para cada gen con la función *p.vector().* Se seleccionan los genes significativos según el p-valor asociado al F-Statistic del modelo.
Esta función tiene dos parámetros:

- *counts*: usa el valor TRUE para análisis de RNA-Seq y FALSE para análisis de microarrays.
- Q: nivel de control del FDR (false discovery rate).

Los outputs de la función son:

- NBp$i → devuelve el numero de genes significativos.
- NBp$alfa → proporciona el p-valor en el control de false discovery *Q.*
- NBp$SELEC → matriz con los genes significativos y sus valores de expresión.

El valor de Q permite aumentar o disminuir la sensibilidad para la selección de los genes significativos. Q ha de seleccionarse según los datos a los que se aplique y según el número de clusters (k) del parámetro de la función see.genes(). Por ello, se va a ejecutar la función para distintos valores de Q y de k. En la imagen a continuación se muestran las líneas del script en R para Q= $10^{-20}$.

```
NBp <- p.vector(matrix_counts, design,Q = 10e-23, counts=TRUE)
NBp$i
```

Se obtienen 7286 genes en este primer filtro.

Una vez se han encontrado genes significativos, se aplica la función *T.fit():* una selección de variables para encontrar variables significativas para cada gen utilizando *stepwise regression*. El resultado es una lista. Para cada gen se dan los siguientes valores:

- p-valor de la regresion ANOVA .
- R-squared del modelo.
- p-valor de los coeficientes de regresión de las variables seleccionadas.

```
NBt <- T.fit(NBp)
```

Se van a generar listas de genes de acuerdo con el modo en que se deseen ver los resultados. En este caso, se desea observar una única lista con los genes más significativos. Para ello se utiliza el parámetro *vars="all". Se usa un r squared* de 0,6.

```
get<-get.siggenes(NBt,rsq = 0.6, vars="all")
```

A continuación se obtiene el número final de genes significativos obtenidos y se genera la gráfica de su agrupación según sus perfiles de expresión.

```
nrow(get$sig.genes$sig.profiles)

genesclust<-see.genes(get$sig.genes, show.fit = T, dis =design$dis,
          cluster.method="hclust" ,cluster.data = 1, k = 6)
```

Las dos figuras que se obtienen con la función *see.genes()* son agrupaciones de genes que se comportan de manera similar. La figura 36 informa de la homogeneidad de cada cluster: el eje x es el array de columnas de la matriz de cuentas formado por las 60 condiciones. El eje y indica el valor de expresión.

La figura 37 muestra, para cada cluster, la mediana de la expresión de los genes para cada una de las 5 condiciones (ordenadas de arriba a abajo): control, mosca mock, bacteria mock, mosca infectada y bacteria infectada.

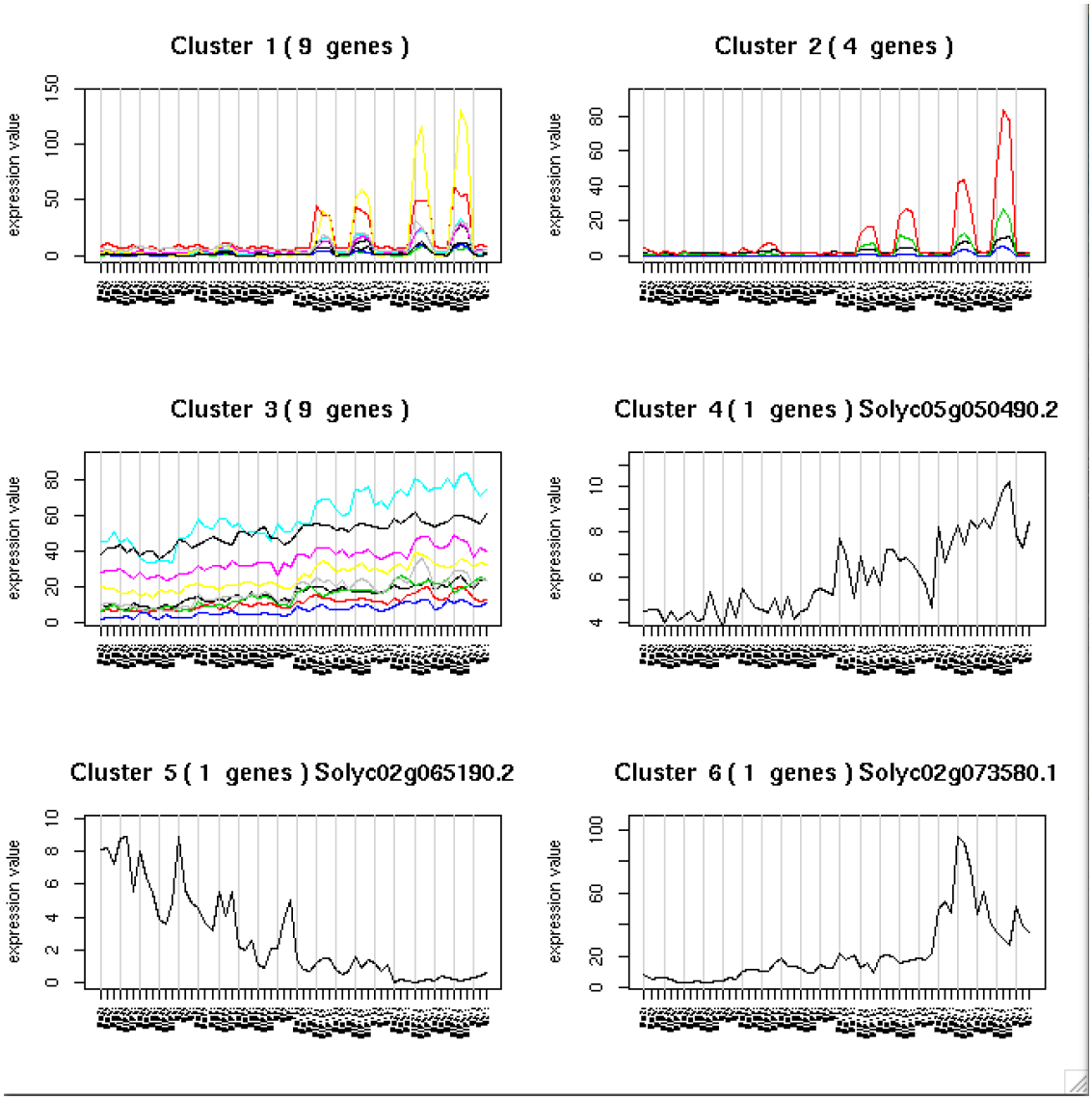



*Figura 36. Gráficas de homogeneidad para cada cluster.*

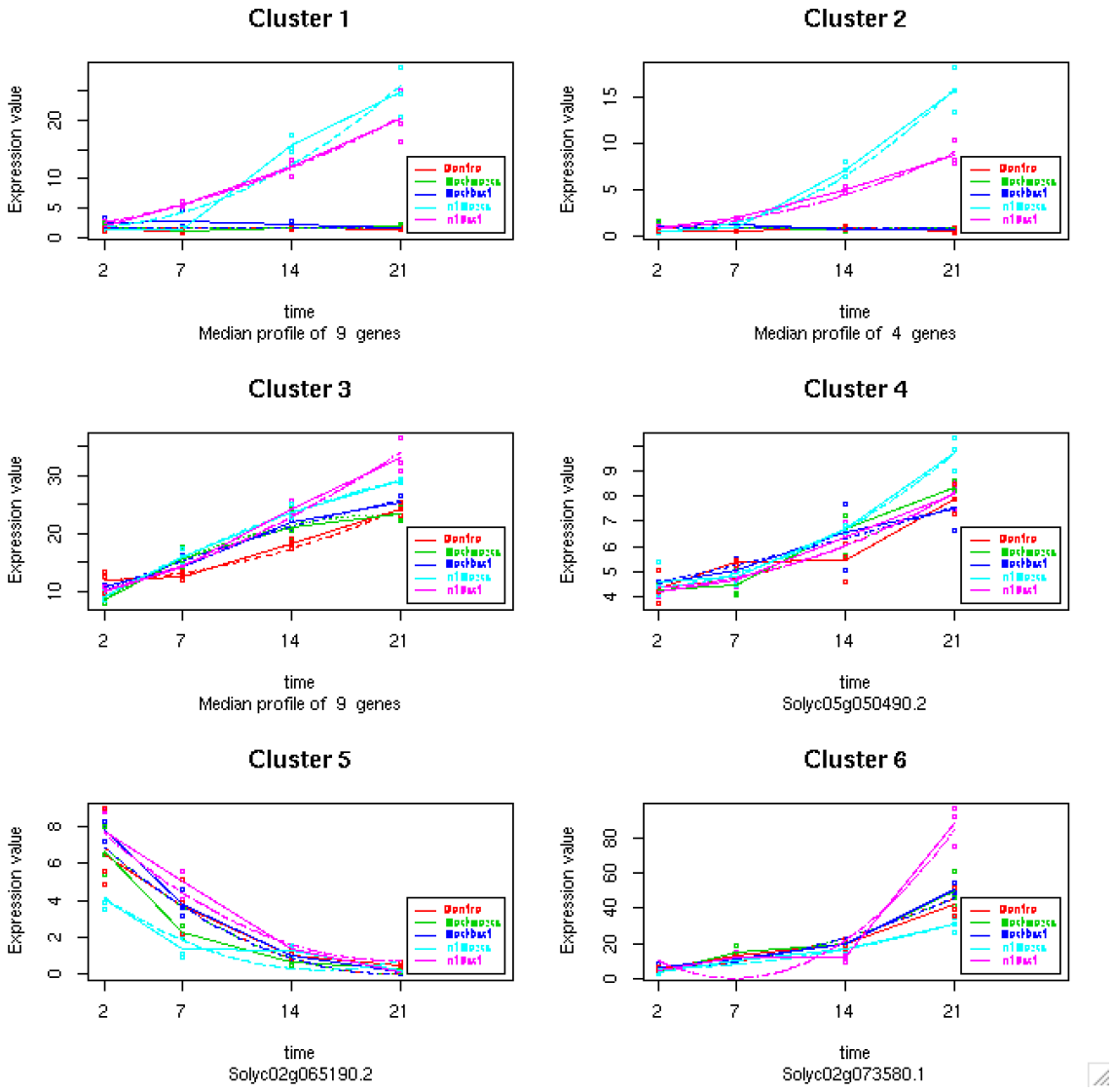



*Figura 37. Gráficas de perfiles medios para cada cluster.*

## *Visualización de agrupaciones para distintas combinaciones de Q y k.*

Las gráficas de la figura 38 muestran clusters en los que sólo hay un gen. El valor de Q es bajo y por ello pocos genes (25) han sido seleccionados. A pesar de disminuir el número de clusters, en la figura 39 también hay clusters con un solo gen.

Un valor menos restrictivo de Q (más alto) puede resultar en más genes agrupados con estos genes únicos y por tanto en nueva información acerca de su corregulación. Para k=6 y Q= $10^{-20}$ (ver figura 40), se ha agrupado un alto número de genes en los clusters 1 y 4, muchos de los cuales no siguen el perfil del cluster y disminuyen la homogeneidad en el que se agrupan. El número de clusters que permite agrupar mejor

los genes es k=9 (ver figura 41), ya que los genes de cada agrupación son realmente similares.

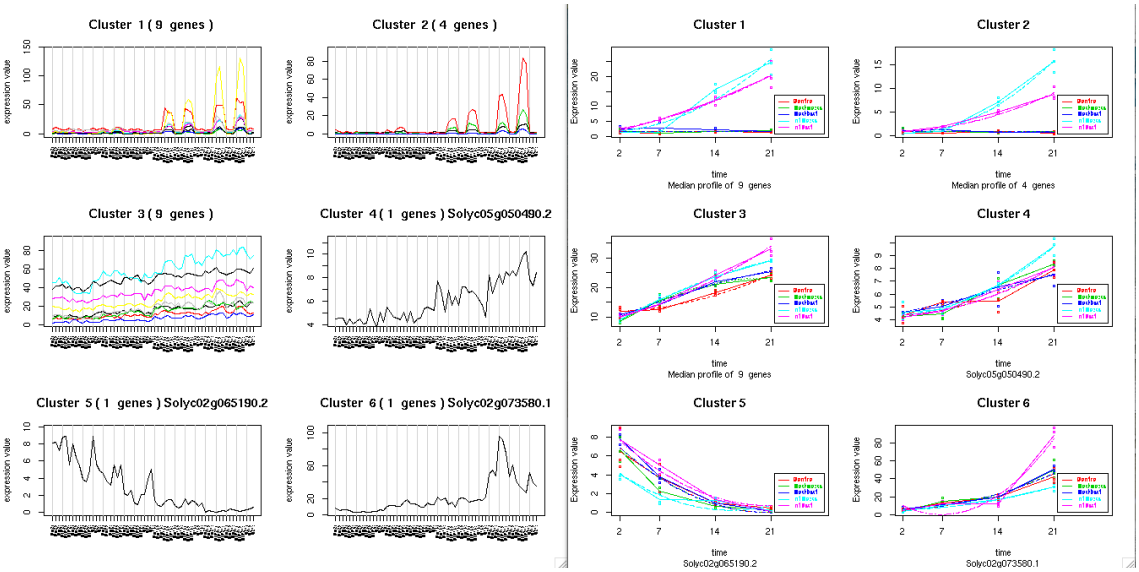



*Figura 38. Resultados para k=6 y Q=$10^{-23}$*

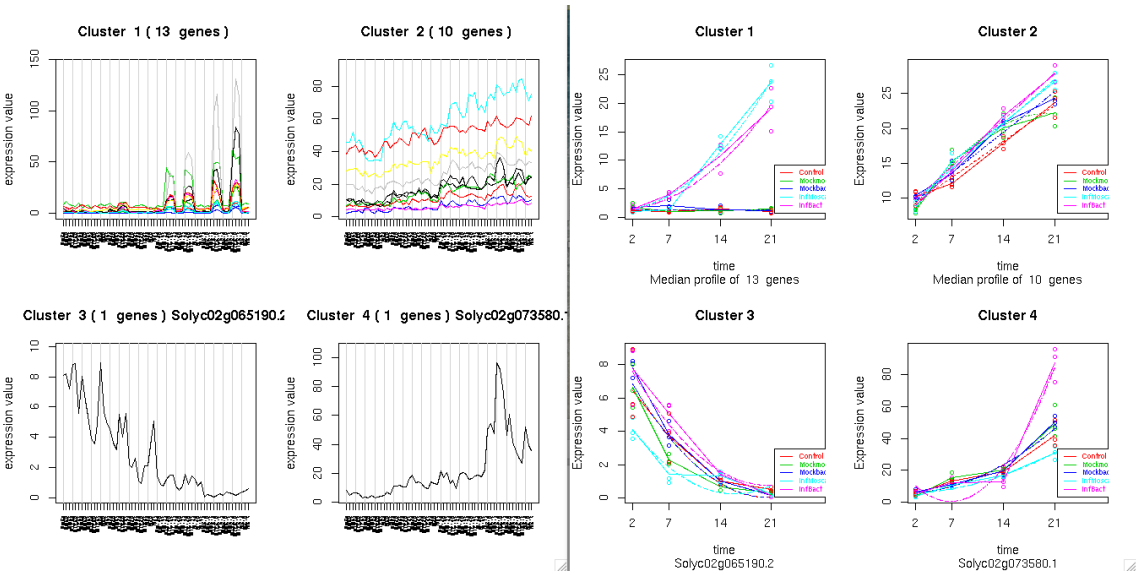



*Figura 39. Resultados para k=4 y Q=$10^{-23}$*

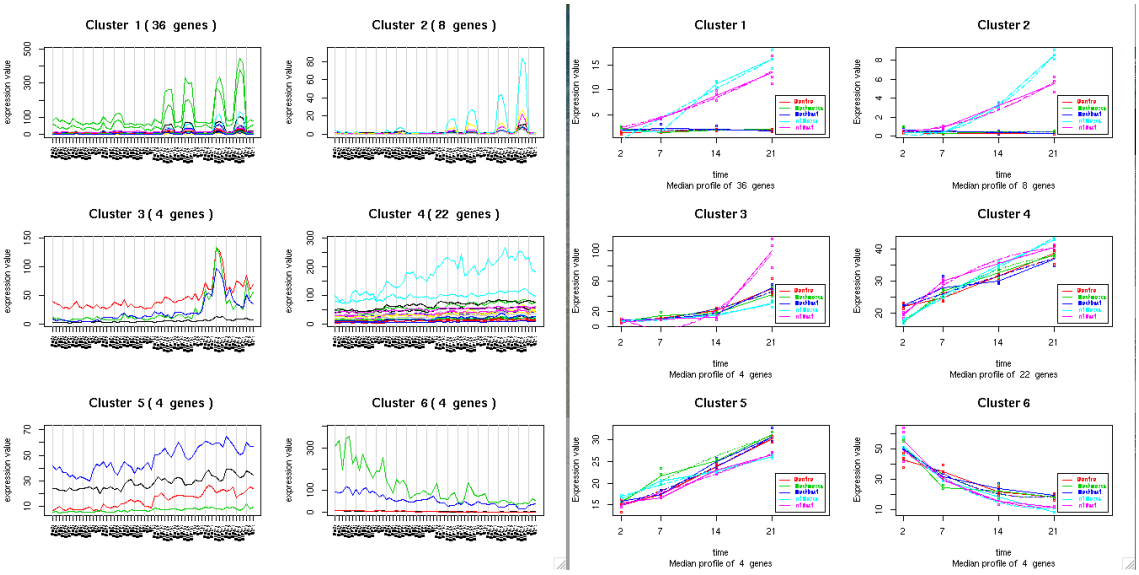



*Figura 40. Resultados para k=6 y Q=$10^{-20}$*

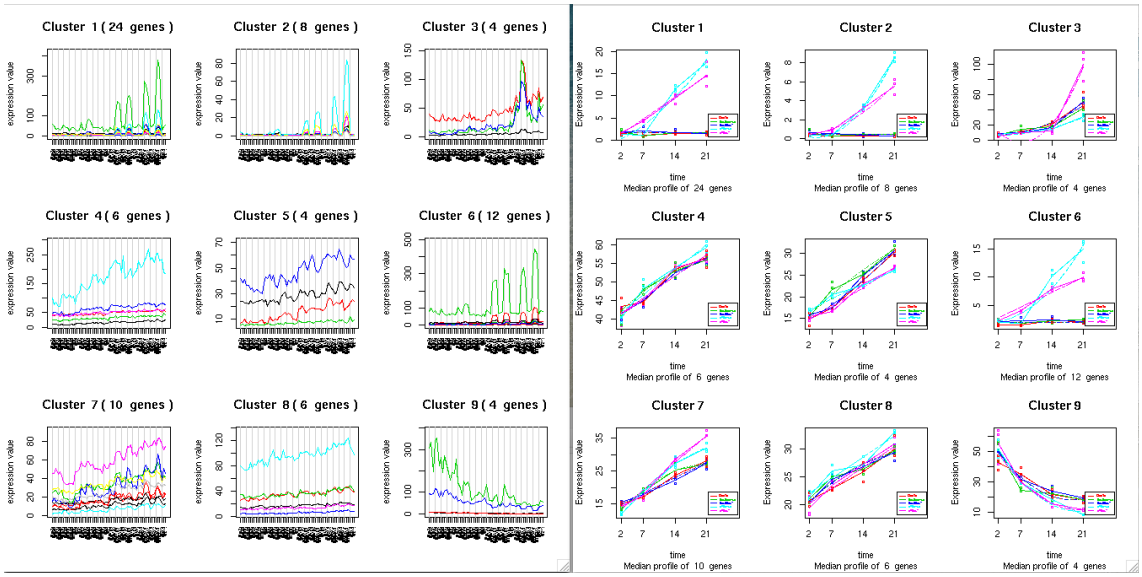



*Figura 41. Resultados para k=9 y Q=$10^{-20}$*

Este tipo de análisis es muy útil para adquirir una visión global del comportamiento de la planta, pero si se desea centrarse en la reacción a la presencia de la mosca infectada y sin infectar, es conveniente observar sólo esas dos condiciones, de manera que las demás condiciones no influyan en el proceso de agrupación.

### *Análisis temporal para dos condiciones*

Se pretende usar esta técnica para observar de forma particular el comportamiento de la planta ante un número menor de condiciones, que son las que nos servirán para responder a las preguntas acerca de la reacción de la planta ante la mosca:

- Planta expuesta a mosca infectada contra planta expuesta a mosca sin infectar
- Planta expuesta a mosca sin infectar contra planta sin tratar o planta control

De la misma forma que se procedió para el análisis bajo multiples condiciones y un control, es posible indicar una única condición y su control. Este tipo de análisis uno a uno se realizará dos veces para las dos combinaciones nombradas anteriormente.

A partir de la matriz de cuentas, se genera una matriz con las columnas de las condiciones que se van a comparar. Este script es el correspondiente a la comparación de planta expuesta a mosca infectada contra planta expuesta a mosca sin infectar.

```
matrix_counts_Mm_MTY<-cbind(matrix_counts[,7],matrix_counts[,8],matrix_counts[,9],matrix_counts[,10],
                            matrix_counts[,11],matrix_counts[,12],matrix_counts[,22],matrix_counts[,23],
                            matrix_counts[,24],matrix_counts[,25],matrix_counts[,26],matrix_counts[,27],
                            matrix_counts[,37],matrix_counts[,38],matrix_counts[,39],matrix_counts[,40],
                            matrix_counts[,41],matrix_counts[,42],matrix_counts[,52],matrix_counts[,53],
                            matrix_counts[,54],matrix_counts[,55],matrix_counts[,56],matrix_counts[,57])
```

Se nombran las columnas de la nueva matriz.

```
colnames(matrix_counts_Mm_MTY)<-c("Mm2a","Mm2b","Mm2c","MTY2a","MTY2b","MTY2c","Mm7a","Mm7b","Mm7c","MTY7a",
                                  "MTY7b","MTY7c","Mm14a","Mm14b","Mm14c","MTY14a","MTY14b","MTY14c","Mm21a",
                                  "Mm21b","Mm21c","MTY21a","MTY21b","MTY21c")
```

Se crea posteriormente la matriz de diseño según la comparación de la que se trate.

```
matrix_design_Mm_MTY<-matrix(nrow=24,ncol=4)
rownames(matrix_design_Mm_MTY)<-as.character(colnames(matrix_counts_Mm_MTY))
colnames(matrix_design_Mm_MTY)<-c("Time","Replicate","Mockmosca","InfMosca")
matrix_design_Mm_MTY[,1]<-c(2,2,2,2,2,2,7,7,7,7,7,7,14,14,14,14,14,14,21,21,21,21,21,21)
matrix_design_Mm_MTY[,2]<-c(1,1,1,2,2,2,3,3,3,4,4,4,5,5,5,6,6,6,7,7,7,8,8,8)
matrix_design_Mm_MTY[,3]<-c(1,1,1,0,0,0,1,1,1,0,0,0,1,1,1,0,0,0,1,1,1,0,0,0)
matrix_design_Mm_MTY[,4]<-c(0,0,0,1,1,1,0,0,0,1,1,1,0,0,0,1,1,1,0,0,0,1,1,1)


design_Mm_MTY <- make.design.matrix(matrix_design_Mm_MTY)
```

Con la función *p.vector()* se seleccionan los genes significativos y con *T.fit()* se buscan variables significativas para cada gen.

```
NBp_Mm_MTY <- p.vector(matrix_counts_Mm_MTY, design_Mm_MTY,
                       Q = 10e-5, counts=TRUE) #keep changing Q to play with number of genes

NBt_Mm_MTY <- T.fit(NBp_Mm_MTY)
```

Posteriormente, se obtienen los genes y se muestran en clústers según sus perfiles de expresión.

```
get_Mm_MTY<-get.siggenes(NBt_Mm_MTY,rsq = 0.6, vars="all")
```

```
genesclust_Mm_MTY<-see.genes(get_Mm_MTY$sig.genes, show.fit = T, dis =design_Mm_MTY$dis,
                             cluster.method="hclust" ,cluster.data = 1, k = 6)
```

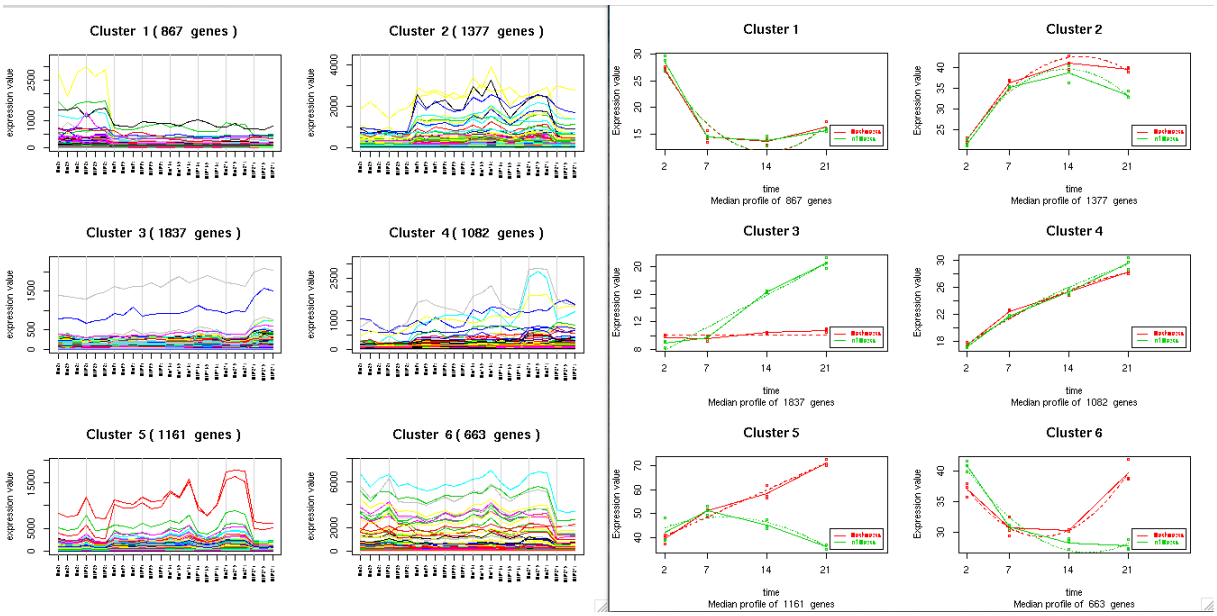



*Figura 42. Resultados para la comparación de condiciones Mock_mosca y Mosca_TYLCV.*

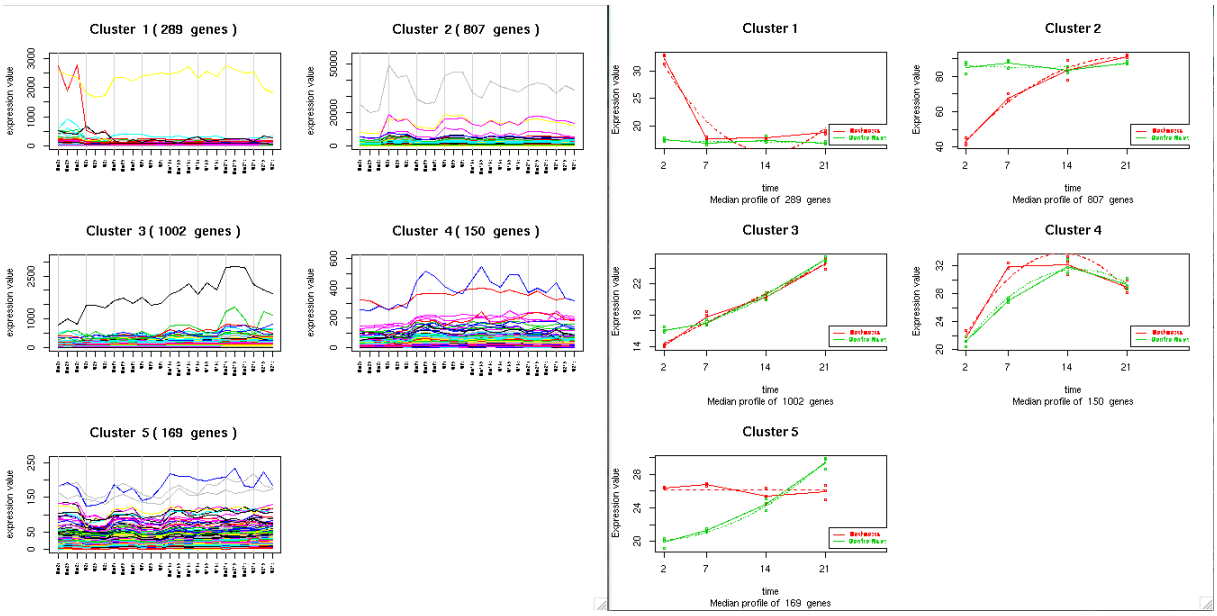



*Figura 43. Resultados para la comparación de condiciones Mock_mosca y Control o naive.*

# 7. Resultados y conclusiones

## Discusión de resultados

En este proyecto se han empleado las tecnologías más utilizadas actualmente, muchas de ellas adaptadas recientemente al NGS (Next Generation Sequencing), como son SeqTrimNext, DESeq2 y maSigPro. Con la intención de reproducir fielmente los resultados del primer análisis, la elección de las herramientas a usar fue también influenciada según las herramientas usadas en dicho análisis.

Estos resultados confirmaron los resultados previos, lo cual permitió seguir avanzando. El análisis temporal de expresión diferencial de la segunda fase de este proyecto proporcionó una gran cantidad de información: genes que se comportan igual, trayectoria de un gen a lo largo del tiempo y a lo largo de diferentes condiciones, posibilidad de observar genes que tienen un comportamiento inverso…

Una de las razones de la decisión de desarrollar un análisis temporal de expresión diferencial fue la necesidad de comprender la variación de la desregulación de los genes de las plantas expuestas a mosca infectada y sana. Una posible explicación a estos resultados es que el comportamiento de la mosca infectada es distinto al de la mosca sin infectar: cuando la Bemisia está infectada por el virus TYLCV, introduce su stiletto con mayor frecuencia que una mosca sana en la planta en busca de floema [25]. Por ello, causa más daño a la planta. Entonces, la diferencia entre la desregulación de los genes de la planta expuesta a una mosca infectada y la de los genes de la planta expuesta a una mosca sana se puede deber a la respuesta de la planta ante el daño de la mosca. Este efecto disminuye con el paso de los días, mientras que la desregulación que se observa a partir de los 14 días sí está relacionada con el virus. De la misma manera, al comparar la planta control (naïve) con la planta expuesta a mosca sana, se puede observar la inicial respuesta a la mosca por parte de la planta y la posterior respuesta a la infección. En este caso, la reacción de defensa ante la mosca es más leve debido a que la mosca está sana y no causa tanto daño como la mosca infectada.

Si se aplica esta teoría a los resultados obtenidos, se puede observar en la figura 44 que son distintos genes los que al principio se desregulan de los que se desregulan al final. En los clusters 1 y 6 se observan los genes que reaccionan a la presencia de la mosca, estando mas desregulados los genes en presencia de la mosca infectada. A los 7 días, los genes para ambas condiciones se desregulan de forma muy similar, con lo que la expresión diferencial entre estas dos condiciones disminuye. A partir de entonces, los genes de la planta expuesta a mosca infectada se desregulan más que los de la planta expuesta a mosca sana, confirmando los números de la tabla de resultados.

Por otro lado, los resultados obtenidos en la comparación de planta control contra planta expuesta a mosca sana se pueden confirmar con la figura 45. Los genes se desregulan en presencia de la mosca hasta pasados los 14 días. Según el cluster en el que se encuentran, los genes se desregulan de diferente forma.

Si se efectuara un enriquecimiento functional, sería de esperar que los genes sobreexpresados al inicio de la exposición de la planta a la mosca estuvieran relacionados con procesos de respuesta a estrés.

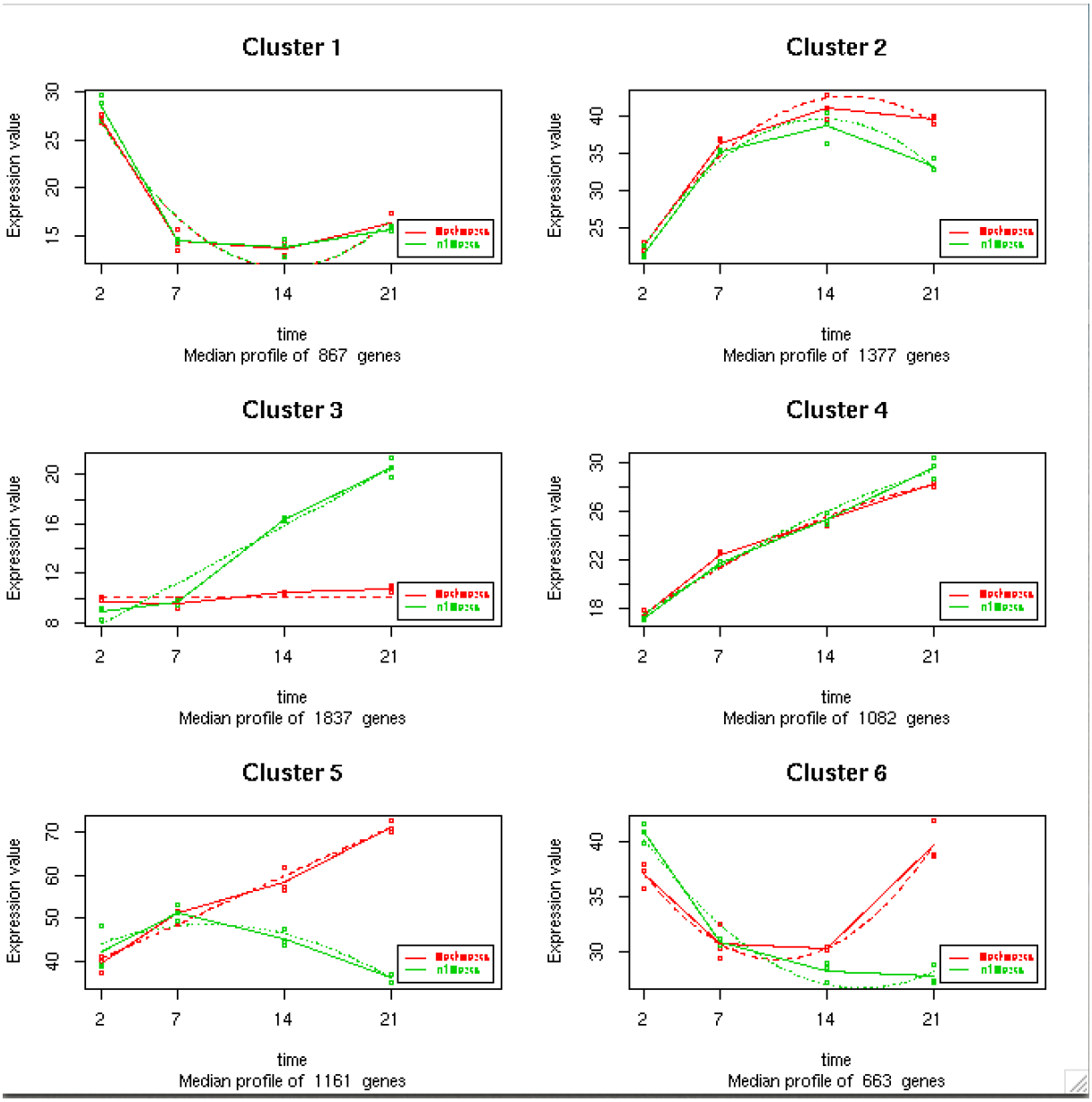



*Figura 44. Mediana de resultados de expresión para condiciones Mock_mosca y Mosca_TYLCV.*

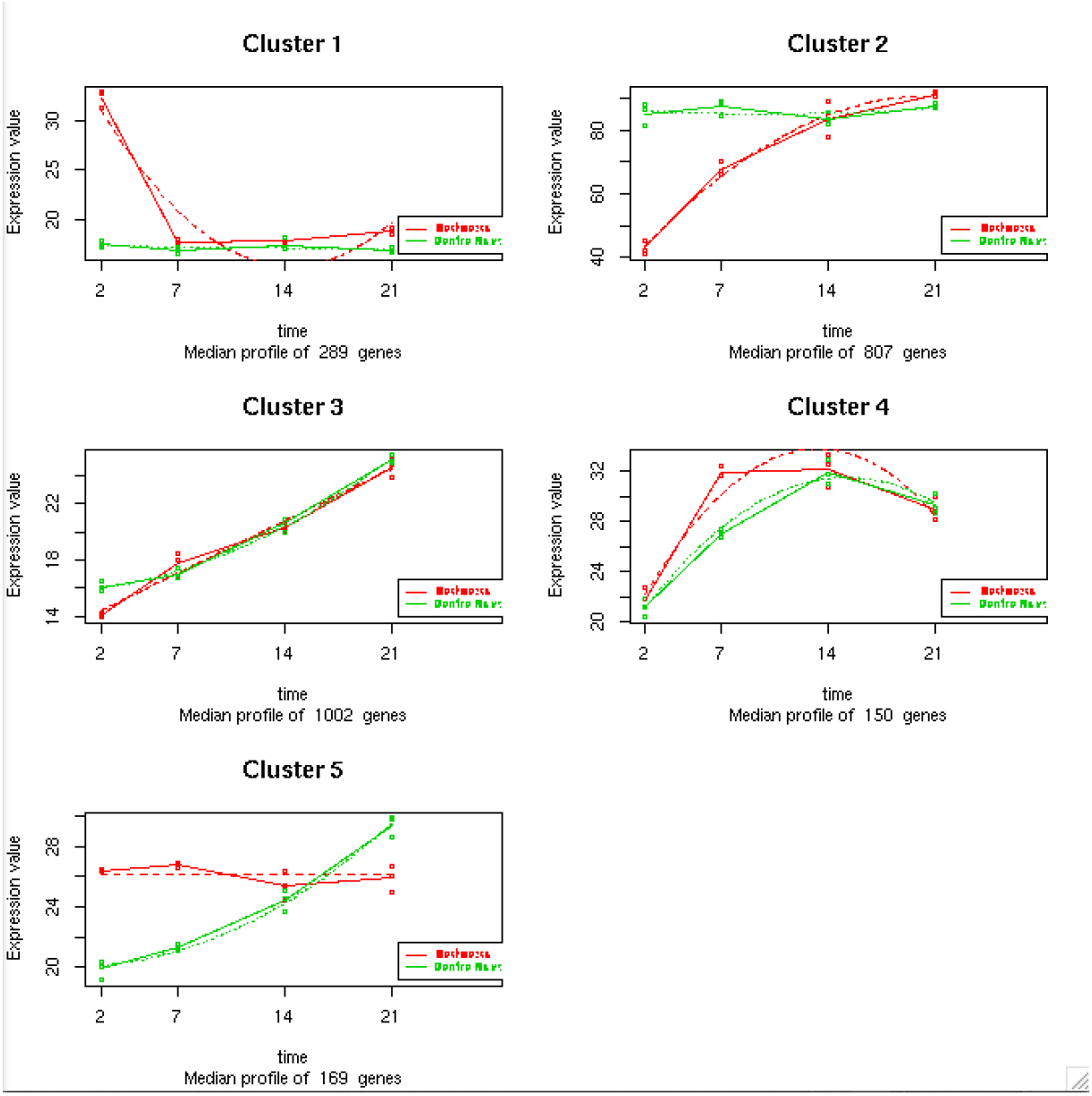



*Figura 45. Mediana de resultados de expresión para condiciones Mock_mosca y Control o naive.*

## Revisión de cumplimiento de objetivos

El motivo principal de este proyecto ha sido estudiar las últimas técnicas de análisis y experimentar con ellas para poner de manifiesto la importancia de un análisis a lo largo del tiempo. Los objetivos propuestos han sido alcanzados aplicando diversas técnicas de bioinformática y aprendiendo nuevas, realizando un análisis de RNA-Seq completo, desde los resultados de secuenciación en crudo. Se ha conseguido así demostrar la utilidad de un análisis temporal de expresión diferencial aplicándolo al problema del virus TYLCV en la planta del tomate.

Entre las dificultades encontradas a lo largo del desarrollo de este proyecto, el mayor desafío fue la familiarización con el funcionamiento del paquete maSigPro y la adaptación de éste a las necesidades de este proyecto. Otra dificultad fue aprender a usar un supercomputador como Picasso y a administrar correctamente los recursos que hay disponibles para el usuario. La interpretación de los resultados de significado biológico ha sido un reto superado gracias a los artículos publicados sobre este tema y al debate con profesionales en este campo.

Este proyecto se ha ideado con la intención de implementar muchas de las competencias que un bioinformático debe saber desarrollar: uso de bash y lenguaje R, manejo de grandes volúmenes de datos, la capacidad de ser autodidacta, y la integración de información relacionada con temas biológicos de diversas fuentes científicas.

### Trabajo futuro

Como trabajo futuro, se pretende automatizar el proceso de elección de los parámetros Q (número de genes) y k (número de clústers). Mediante un enriquecimiento functional, se va a obtener un valor de calidad para cada clúster según la relación entre las funciones de los genes en cada uno, tomando este valor como muestra de calidad para cada combinación de Q y k.

Este método de análisis supone una herramienta imprescindible para comprender la interacción virus-huésped, pero también la interacción de ambos con el vector de transmisión y la dinámica subyacente no directamente relacionada con el virus.

## 8. Referencias bibliográficas

A continuación se cita la bibliografía utilizada para desarrollar este proyecto.

*[1] Wenxi Ning et al. Transmission of Tomato Yellow Leaf Curl Virus by Bemisia tabaci as Affected by Whitefly Sex and Biotype (2015) Scientific Reports. doi:10.1038/srep10744*

*[2] Guillermo R Et al.,(2012). Characterization of the Arabidopsis Thaliana interactome targeted by viruses. ResearchGate.*

*[3] Tomato yellow leaf curl virus alters the host preferences of its vector Bemisia tabaco. Scientific Reports 3, Article number: 2876 (2013) doi:10.1038/srep02876*

*[4] Encyclopedia of virology (3rd edition), (2008). Pages 138-145.*

*[5] Qi Su, Mark C. Mescher, Shaoli Wang, Gong Chen, Wen Xie,… & Youjun Zhang. Tomato yellow leaf curl virus differentially influences plant defence responses to a vector and a non-vector herbivore.(2016). DOI: 10.1111/pce.12650*

*[6] Jolly Basak. Tomato Yellow Leaf Curl Virus: A Serious Threat to Tomato Plants World Wide. 2016 DOI: 10.4172/2157-7471.1000346*

*[7] M José Nueda,(2014). Next maSigPro: updating maSigPro bioconductor package for RNA-seq time series. iSCB.*

*[8] Statistic Solutions, (2013). [ONLINE] . Available at: http://www.statisticssolutions.com/manova-analysis-anova/*

*[9] Kim, Jaehee,(2013) .A method to identify differential expression profiles of time-course gene data with Fourier transformation. Biomed Central.*

*[10] SCBI Documentation. [ONLINE] . http://www.scbi.uma.es/site/scbi/documentation*

*[11] Falgueras J1, Lara AJ, Fernández-Pozo N, Cantón FR, Pérez-Trabado G, Claros,(2010). MG.SeqTrim: a high-throughput pipeline for pre-processing any type of sequence read. BMC Bioinformatics. doi: 10.1186/1471-2105-11-38.*

*[12] Metzker, M. L. (2010). Sequencing technologies—the next generation. Nature reviews genetics, 11(1), 31-46.*

*[13] Dobin A1, Davis CA, Schlesinger F, Drenkow J, Zaleski C, Jha S, Batut P, Chaisson M, Gingeras TR. (2013). STAR: ultrafast universal RNA-seq aligner. Bioinformatics. doi: 10.1093/bioinformatics/bts635.*

*[14] Julio Sergio Santana et al., (2014). El arte de programar en R: un lenguaje para la estadística.*

*[15] Ihaka, R., & Gentleman, R. (1993). R project. [ONLINE] http://www.r-project.org.*

*[16] Racine, J. S. (2012). RStudio: A Platform-Independent IDE for R and Sweave.*

*[17] Gentleman, R C., Carey, V. J., Bates, D. M., Bolstad, B., Dettling, M.... & Hornik, K. (2004) Bioconductor: open software development for computational biology and bioinformatics. Genome biology.*

*[18] Love MI, Huber W and Anders S (2014). "Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2." Genome Biology, 15, pp. 550. doi: 10.1186/s13059-014-0550-8.*

*[19] Conesa A and Nueda MJ (2017). maSigPro: Significant Gene Expression Profile Differences in Time Course Gene Expression Data. R package version 1.48.0, http://bioinfo.cipf.es/.*

*[20] Yates, A., Akanni, W., Amode, M. R., Barrell, D., Billis, K., Carvalho-Silva, D., ...& Girón, C. G. (2016). Ensembl 2016. Nucleic acids research,44(D1), D710-D716.*

*[21] About GTF and GFF format. [ONLINE] http://www.ensembl.org/info/website/upload/gff.html*

*[22] Web: https://www.biostars.org/p/8454/*

*[23] Alexander Dobin and Thomas R. Gingeras (2015). Mapping RNA-seq Reads with STAR. doi: 10.1002/0471250953.bi1114s51*

*[24] Michael I. Love, Simon Anders, and Wolfgang Huber (2017) Analyzing RNA-seq data with DESeq2*

*[25] Moreno-Delafuente A, Garzo E, Moreno A, Fereres A (2013) A Plant Virus Manipulates the Behavior of Its Whitefly Vector to Enhance Its Transmission Efficiency and Spread. PLoS ONE 8(4): e61543. doi:10.1371/journal.pone.0061543*